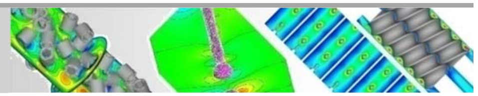
Hardware-oriented Numerics for PDEs

Motivation, Concepts, Applications

Stefan Turek, Dominik Göddeke
Institut für Angewandte Mathematik , LS III
Technische Universität Dortmund
ture@featflow.de

http://www.mathematik.tu-dortmund.de/LS3
http://www.featflow.de

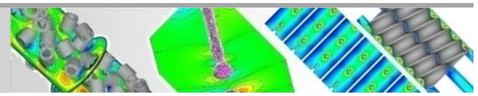


Hardware-oriented Numerics for PDEs

Motivation, Concepts, Applications

This talk will provide a motivation for HWON, shares general ideas regarding algorithmic, numerical and computational challenges und demonstrates exemplarily the application onto multiphase flow problems.

For mathematical and algorithmic details, particularly w.r.t. GPU Computing, please join the corresponding Minisymposium (after this talk.....☺)



Motivation: “Hardware isn’t our friend anymore….”

I) Scientific Computing faces a paradigm shift

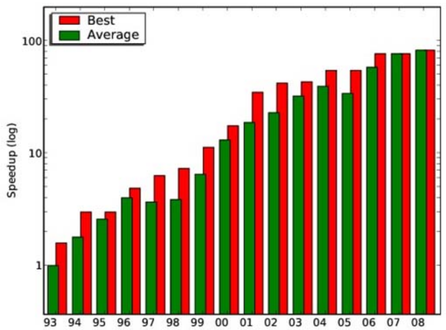
II) Unconventional hardware has to be taken into account

III) Realistic applications: *Virtual Labs* for Multiphase flow

Motivation: “Hardware isn’t our friend anymore….”

I) Scientific Computing faces a paradigm shift

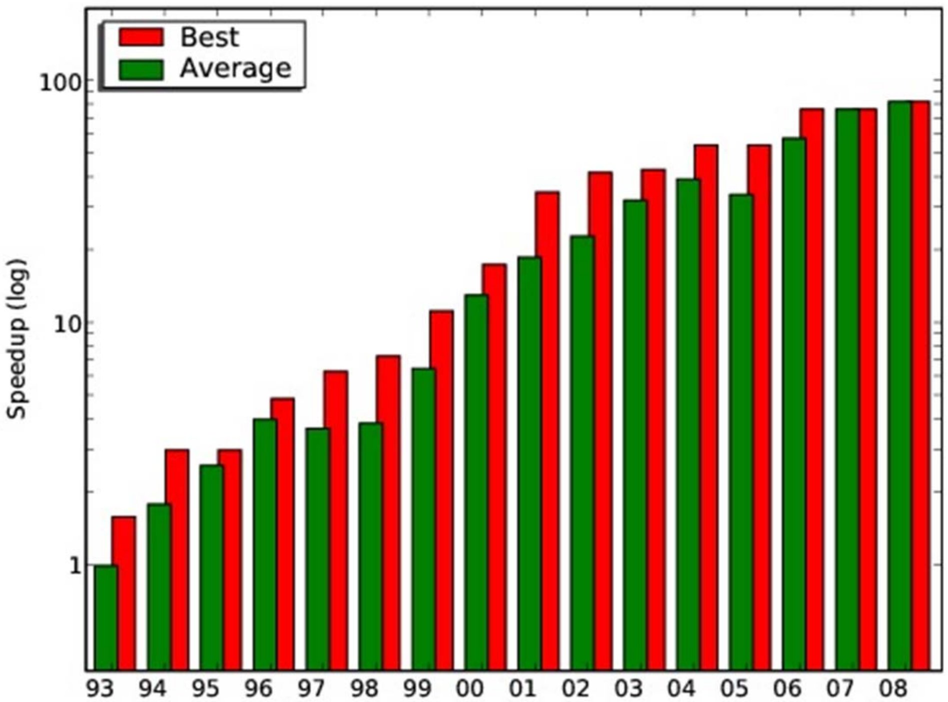
– Adaptive Finite Element Methods (AFEM) and Multigrid Solvers: most flexible, efficient and accurate simulation tools for PDEs nowadays, but **software realization no longer runs faster automatically on newer hardware**



Motivation: “Hardware isn’t our friend anymore….”

FeatFlow-Benchmark 1993-2008: FEM-MG code

Speed-up (logscale)



→ Speed-up of 80x for free in 16 years

→ Stagnation for standard simulation tools

→ Absolute performance?

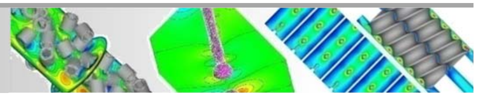
Motivation: “Hardware isn’t our friend anymore….”

I) Scientific Computing faces a paradigm shift

- Adaptive Finite Element Methods (AFEM) and Multigrid Solvers: most flexible, efficient and accurate simulation tools for PDEs nowadays, but **software realization no longer runs faster automatically on newer hardware**

- Single CPU cores are not getting so much faster, while significant speed-up is obtained only via different levels of parallelism

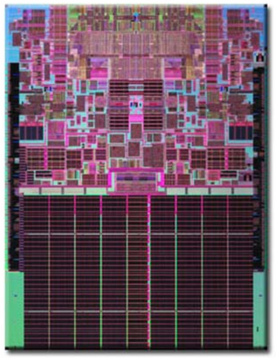
- Data movement gets more expensive due to memory wall (in particular for sparse Linear Algebra problems)



Motivation: “Hardware isn’t our friend anymore....”

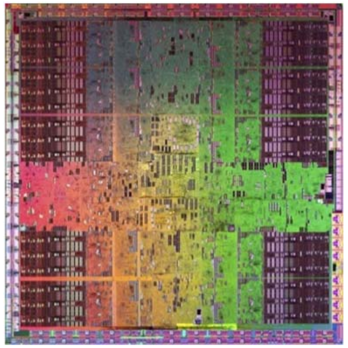
II) *Unconventional* hardware has to be taken into account

- Multicore CPUs - [Cell BE processor (PS3)] - graphics cards (GPUs)
- [HPC accelerators (ClearSpeed)] - reconfigurable hardware (FPGAs)
- Parallelism and heterogeneity everywhere (from single chip in laptops to workstations up to big clusters and supercomputers)
- However: Compilers and libraries are limited



CPUs minimise latency of individual operations with cache hierarchies due to memory wall problem

GPUs maximise throughput over latency and exploit data-parallelism



Motivation: “Hardware isn’t our friend anymore….”

- CELL multicore processor (PS3):
  7 synergistic processing units
  @ 3.2 GHz ≈ 218 GFLOP/s,
  Memory @ 3.2 GHz

- *GPU (NVIDIA GTX 580):*
  512cores @ 1.5 GHz,
  2 GHz memory bus (192 GB/s)
  ≈ 1.6 TFLOP/s

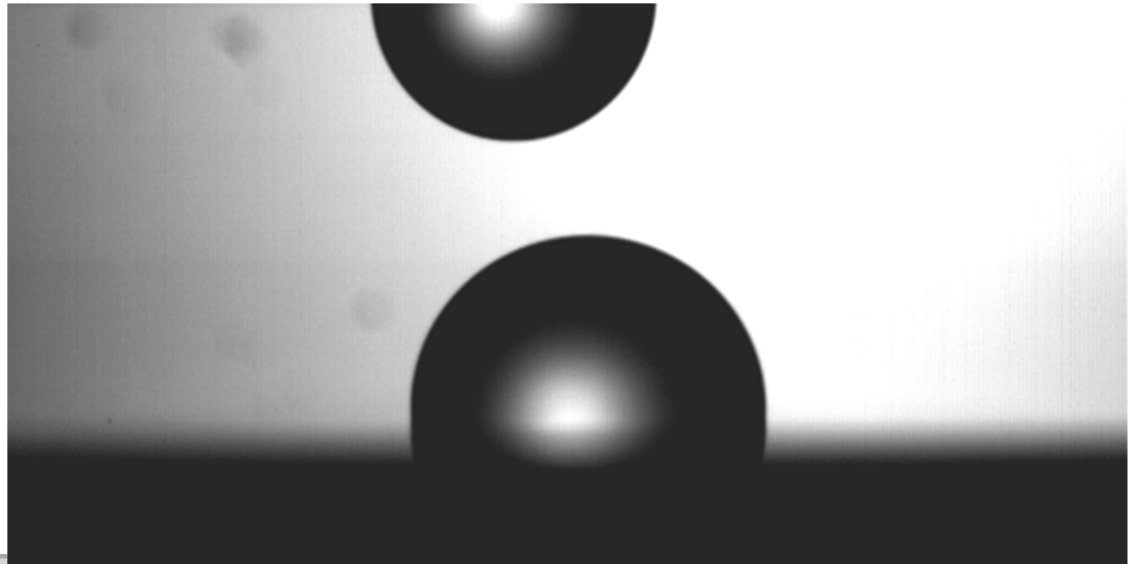
Many papers claim speedups of 100x……Myths vs. Reality

(also multicore CPUs are fast; double vs. single precision; more carefully tuned GPU codes; different numerical efficiency; GPUs as coprocessor for CPUs; …)

Motivation: "Hardware isn't our friend anymore…."

III) Realistic applications: *Virtual Labs* for Multiphase flow

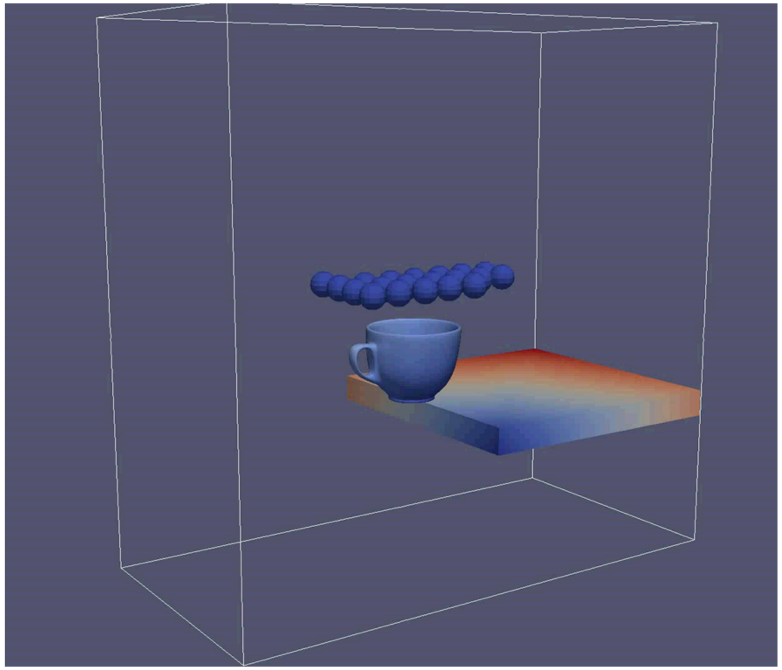
- How to design algorithms and software on these architectures for complete *Virtual Labs* for realistic applications?
- Vision: *Highly efficient, flexible and accurate „real life" simulation based on modern Numerics and algorithms while exploiting modern hardware!*
- Here: Multiphase-CFD as prototype for complex problems

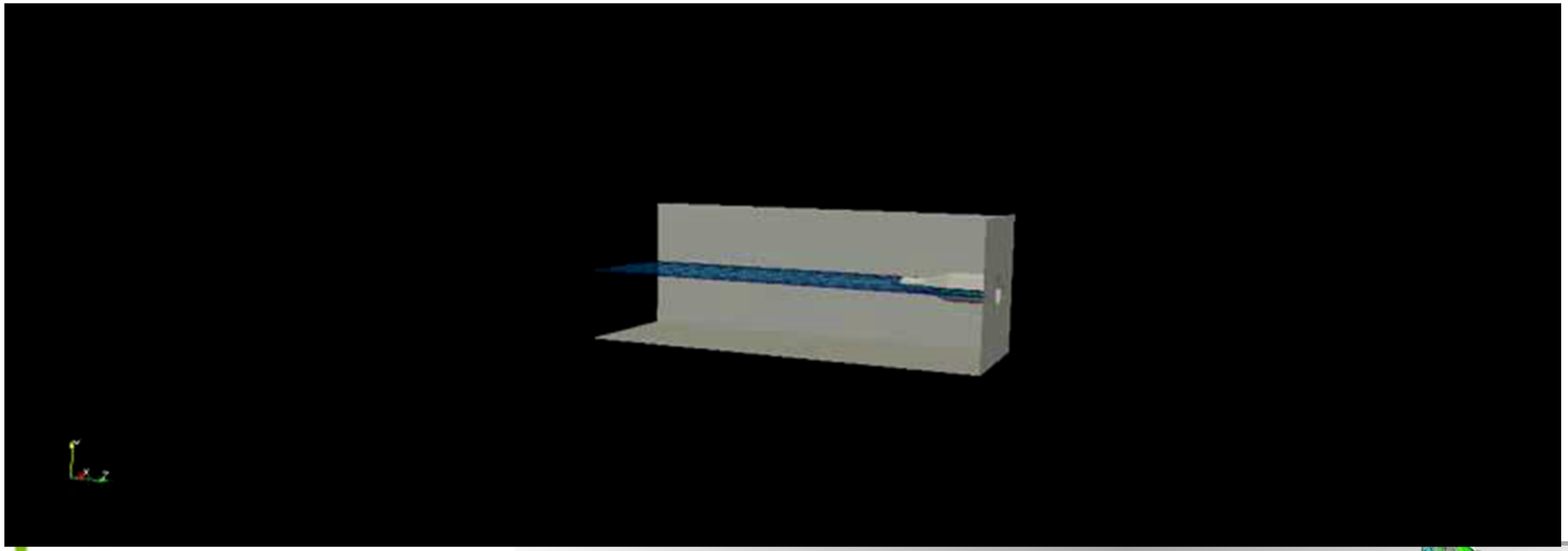


Why Multiphase Problems?

Accurate, robust, flexible and efficient simulation of multiphase problems with dynamic interfaces and complex geometries, particularly in 3D, is still a challenge!

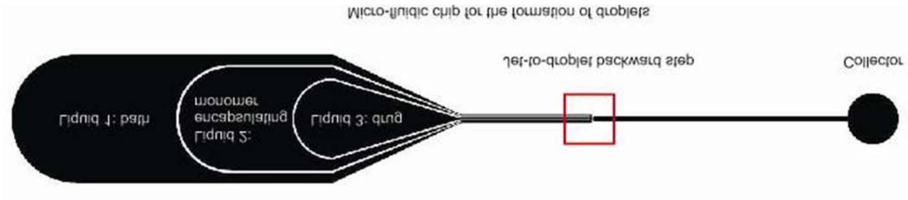
- Mathematical Modelling
- Numerics / CFD Techniques
- Validation / Benchmarking
- HPC Techniques / Software

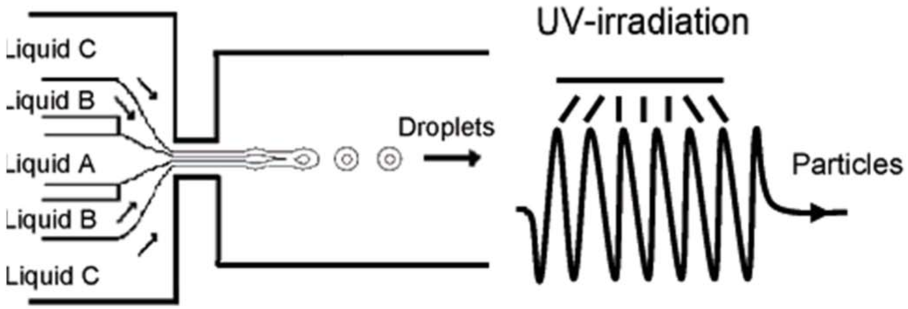


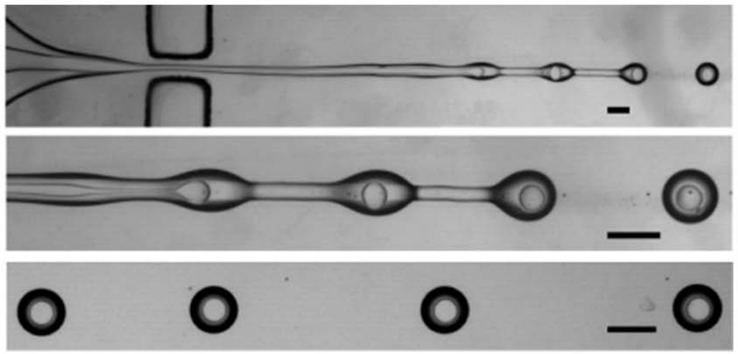


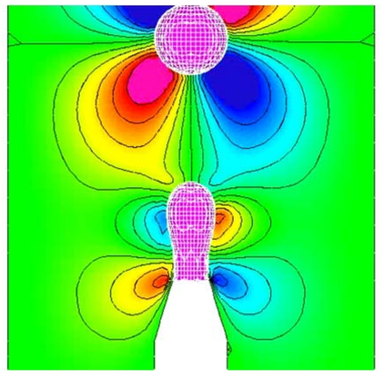
Application I: Micro-fluidic Drug Encapsulation

- Numerical simulation of *drug encapsulation* *("particles in monodisperse compound droplets")* for application in biomedical devices
- Polymeric "bio-degradable" outer fluid with viscoelastic effects
- Optimization of chip design w.r.t. flow rates, droplet size, geometry



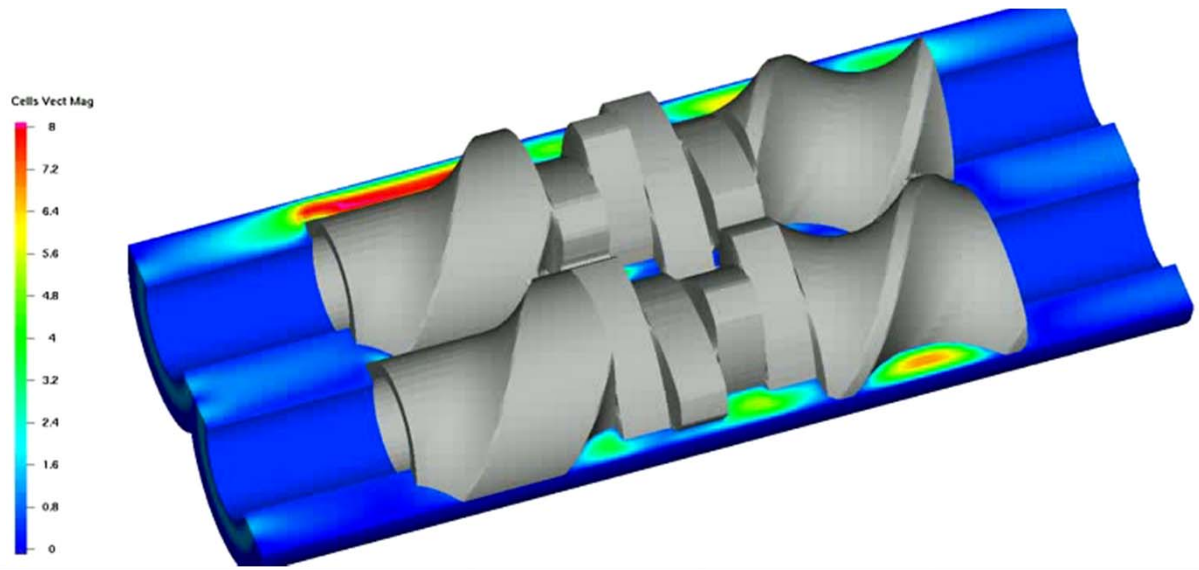


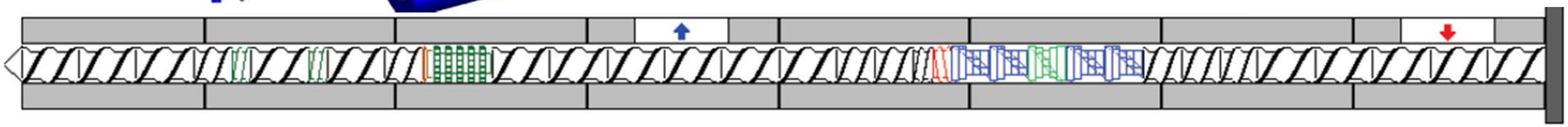




Application II: Twinscrew Extruders

- *Non-Newtonian rheological* models (shear & temperature dependent) with *non-isothermal* flow conditions (cooling from outside, heat production) and solid (granular) particles
- Evaluation of torque acting on the screws, energy consumption
- Prediction of hotspots and maximum shear rates





Aim of this Talk

High Performance Computing

meets

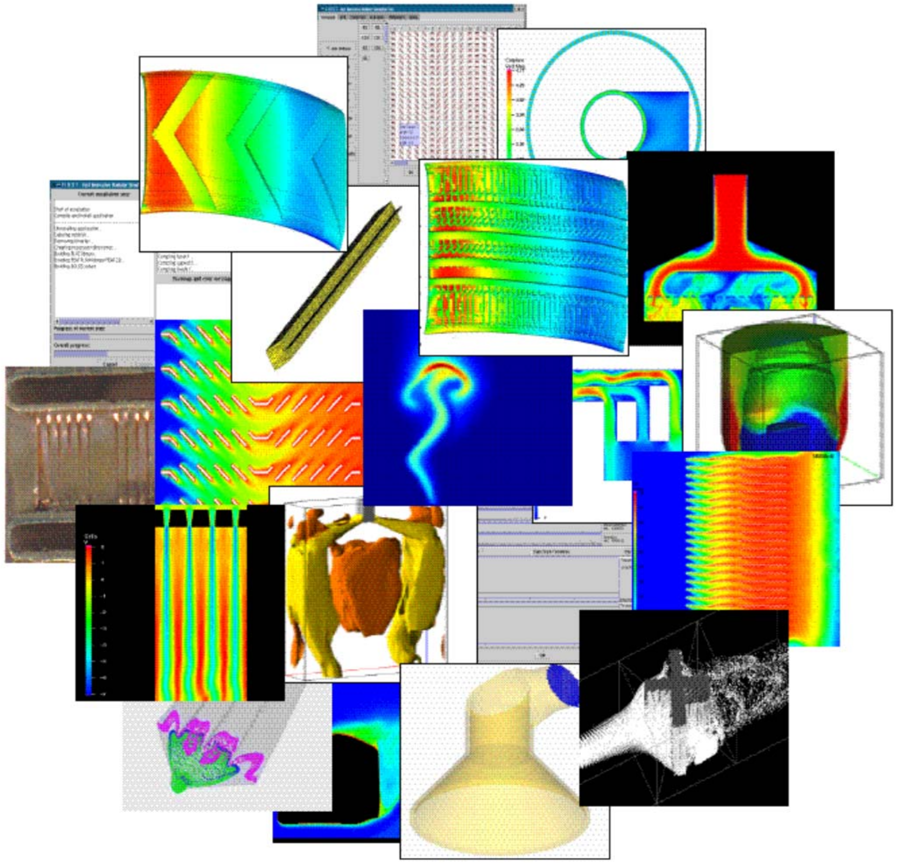
Hardware-oriented Numerics

on

Unconventional Hardware

for

Multiphase Flow Problems



Hardware-Oriented Numerics (HWON)

Use the "best" numerical & algorithmic concepts while exploiting modern hardware <u>at the same time</u>!

- It is more than '*good Numerics*' and '*good Implementation*' on modern (parallel) hardware architecture
- Consider '*short-term hardware developments*' now, but '*long-term hardware trends*' for designing efficient numerical schemes
- 'Total Numerical Efficiency' as critical quantity for balancing numerical efficiency vs. hardware efficiency

Criterion: `Total Numerical Efficiency’

FEM Multigrid solvers with **adaptive meshing** are candidates

- ‘**High** (guaranteed) **accuracy** for user-specific quantities with minimal #d.o.f. (~ *N*) via **fast and robust solvers** – for a wide class of parameter variations – with **optimal numerical complexity** (~ *O*(*N*)) …
  But: while exploiting a significant percentage of the **available huge sequential/ parallel GFLOP/s rates** at the same time’

- What does this mean: Is it easy to achieve high ‘Total Numerical Efficiency’? How to measure?

Example: Fast Poisson Solvers (after FEM discr.)

- ‘Optimized’ Multigrid methods for **scalar PDE** problems (≈Poisson problems) on **general meshes** should require ca. 1000 FLOPs per unknown (in contrast to single-grid Krylov-space methods or direct solvers a la UMFPACK)

- Problem size 10^6 : Much less than 1 sec on PC (???)
- Problem size 10^{12}: Less than 1 sec on PFLOP/s computer

➔ **More realistic (and much harder) ‘Criterion’ for Petascale Computing in Technical Simulations**

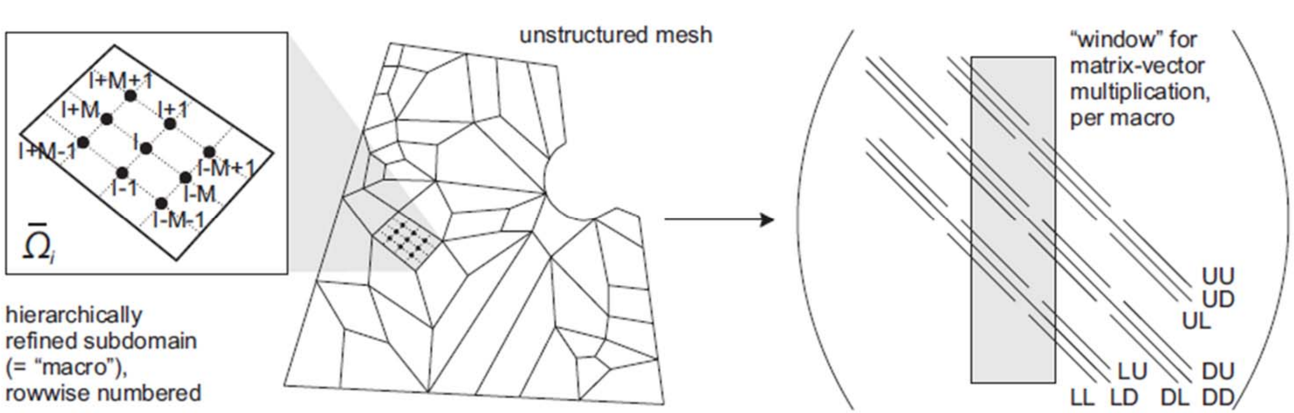
Main Component: ‘Sparse’ MV

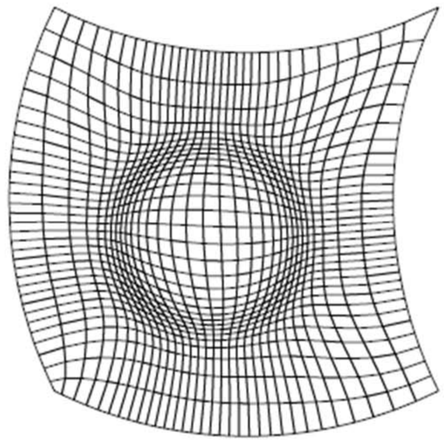
- Sparse Matrix-Vector techniques (‘indexed DAXPY’) on **general unstructured grids**

```
    DO 10 IROW=1,N
       DO 10 ICOL=KLD(IROW),KLD(IROW+1)-1
10        Y(IROW)=DA(ICOL)*X(KCOL(ICOL))+Y(IROW)
```

Fully adaptive grids
Maximum flexibility
‘Stochastic’ numbering
Unstructured sparse matrices
Indirect addressing, very slow.

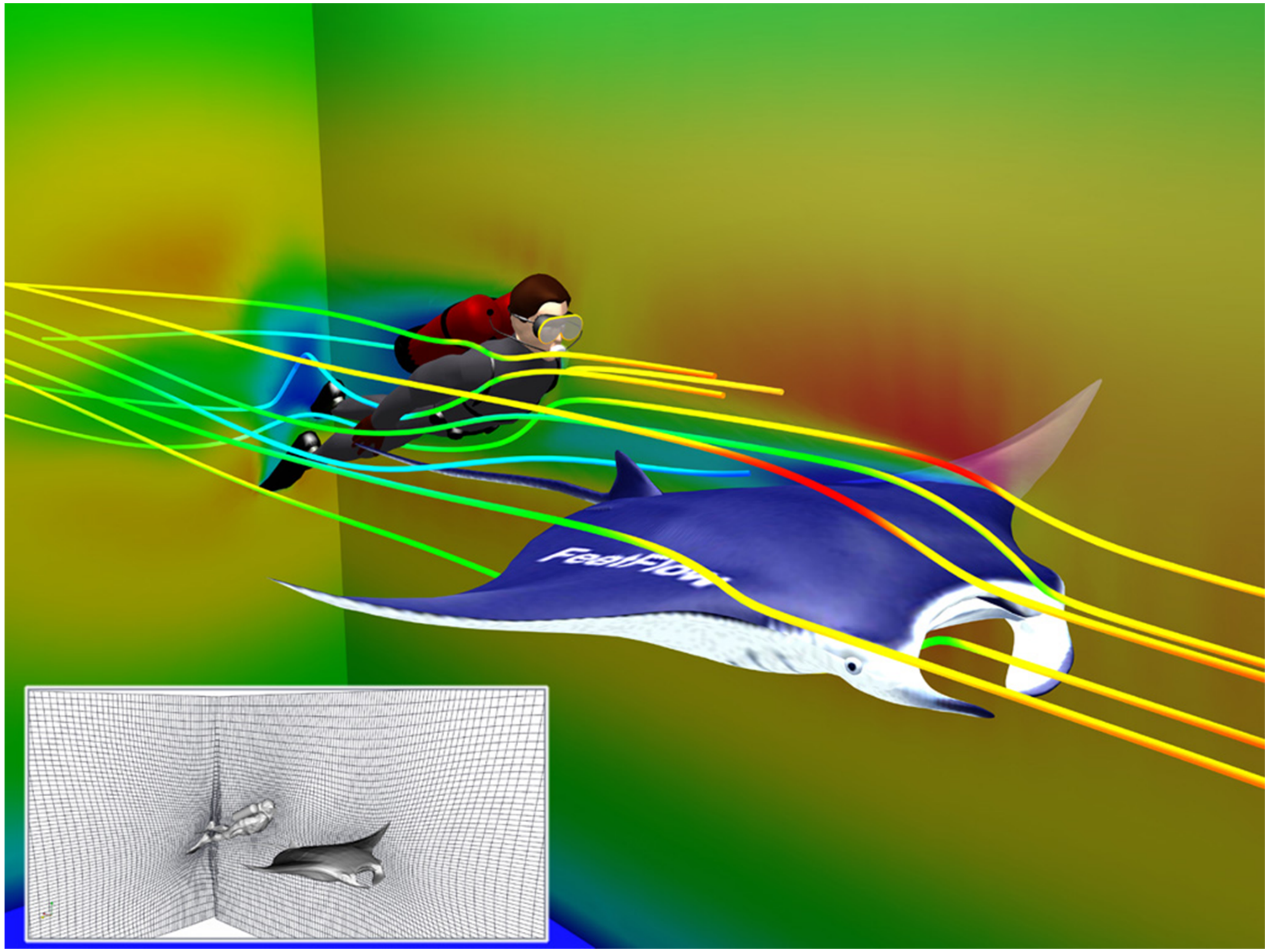
- Sparse Banded Matrix-Vector techniques on **generalized TP grids**





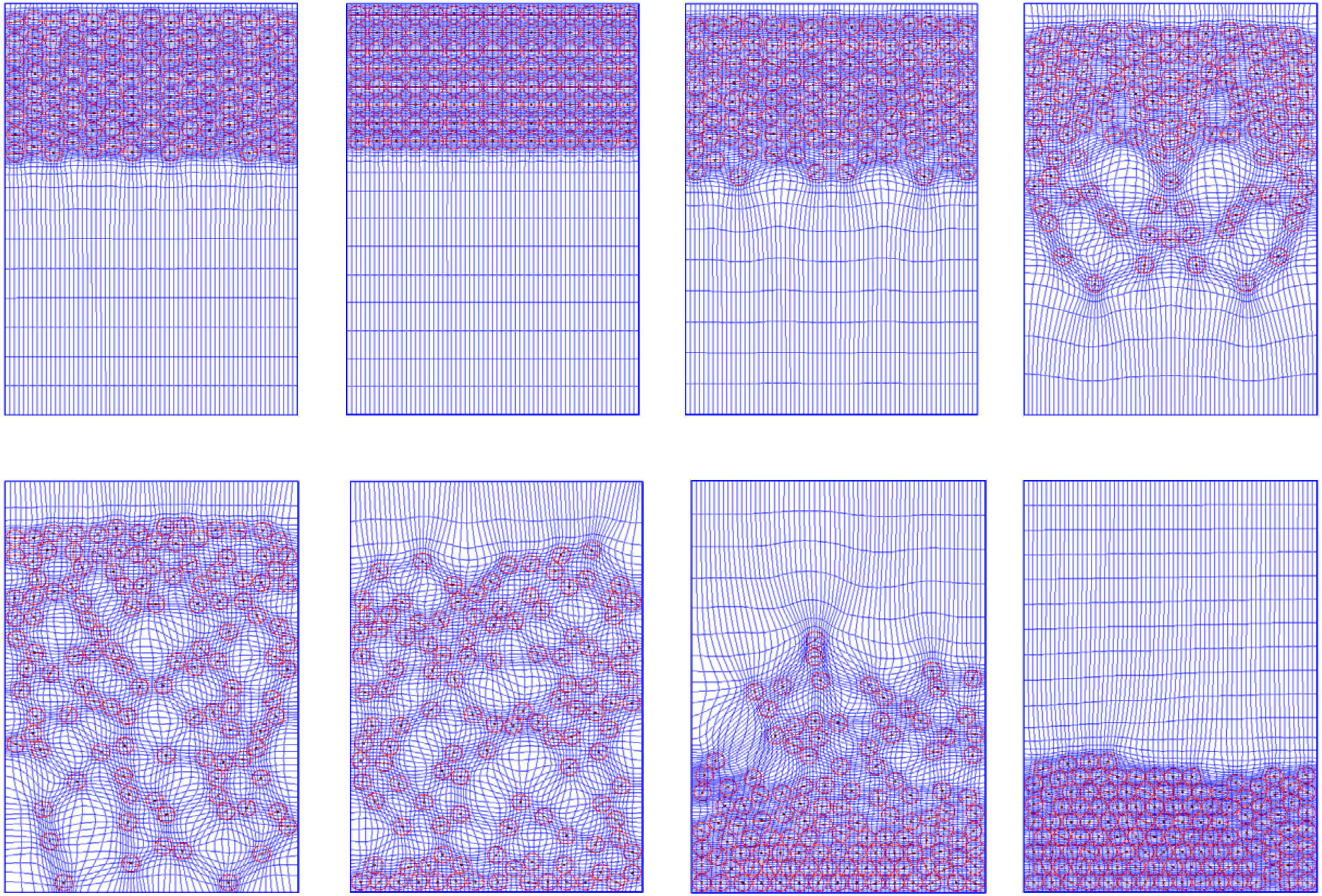
Locally structured grids
Logical tensor product
Fixed banded matrix structure
Direct addressing (⇒ fast)
r-adaptivity

Generalized Tensorproduct Meshes

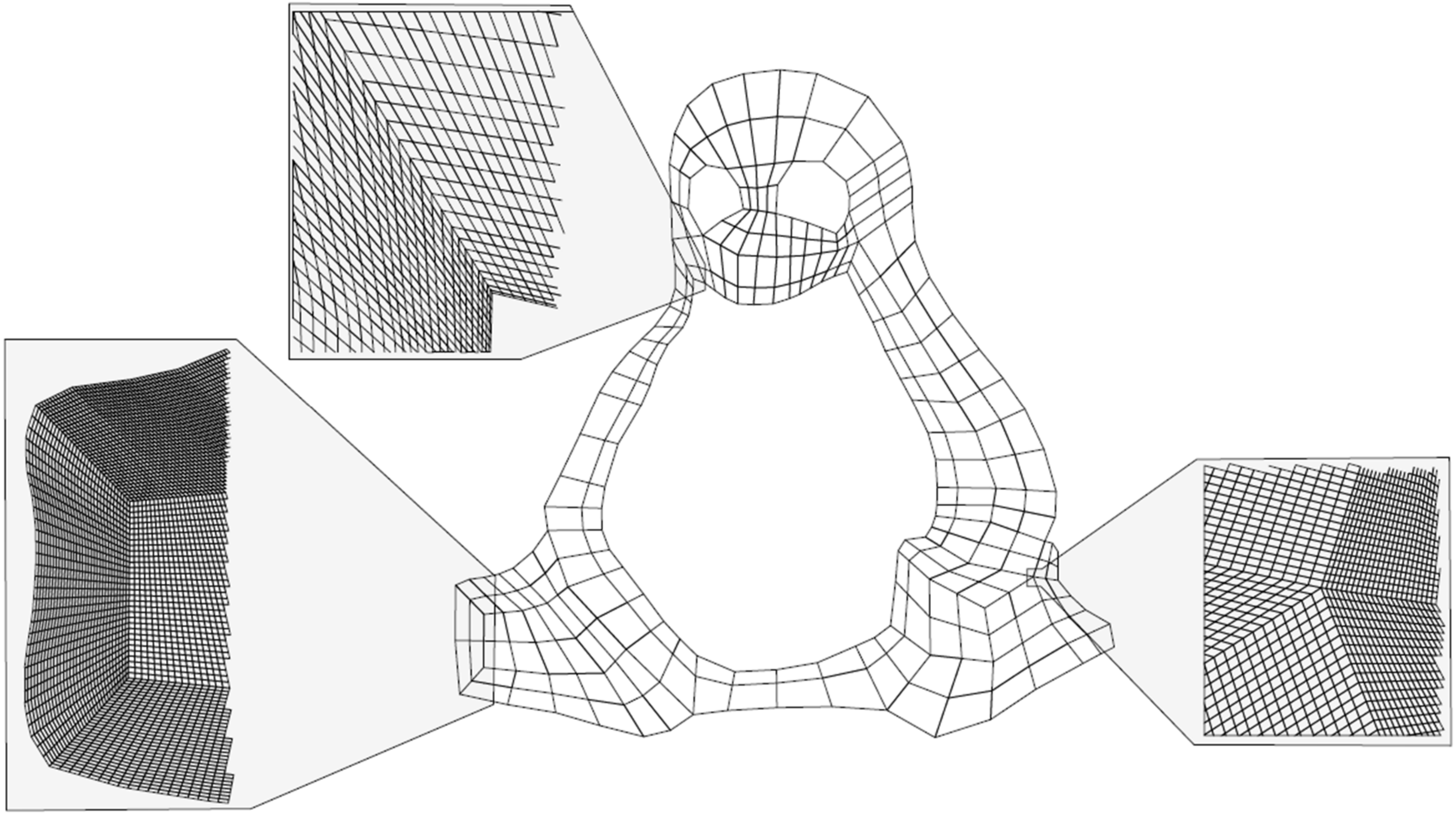


…with **Fictitious Boundary Methods (FBM)** for complex objects

Generalized Tensorproduct Meshes (dynamic)



Generalized Tensorproduct Meshes (piecewise)

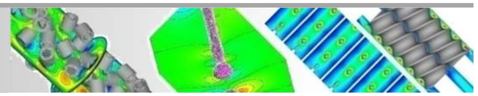


Sparse MV on TP Grids

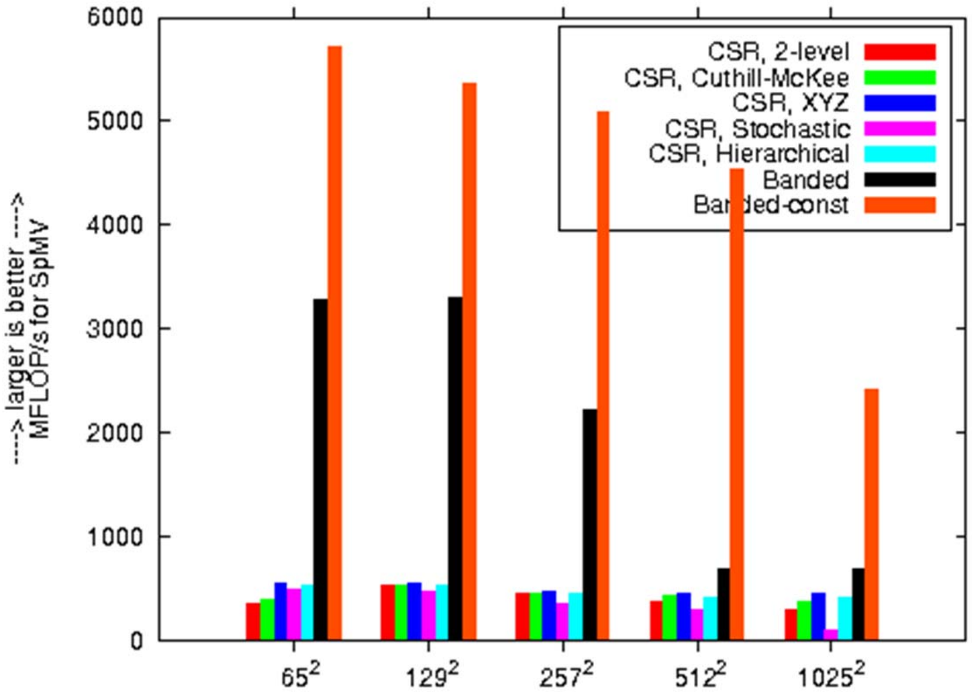
Xeon E5450			
Numbering	**4K DOF**	**66K DOF**	**1M DOF**
Stochastic (CSR)	500	364	**95**
Hierarchical (CSR)	536	445	**418**
Banded	3285	2219	**687**
Stencil (const)	5720	5094	**2415**

In realistic scenarios, MFLOP/s rates for sparse MV are

- often poor, and
- problem size, and
- numbering dependent

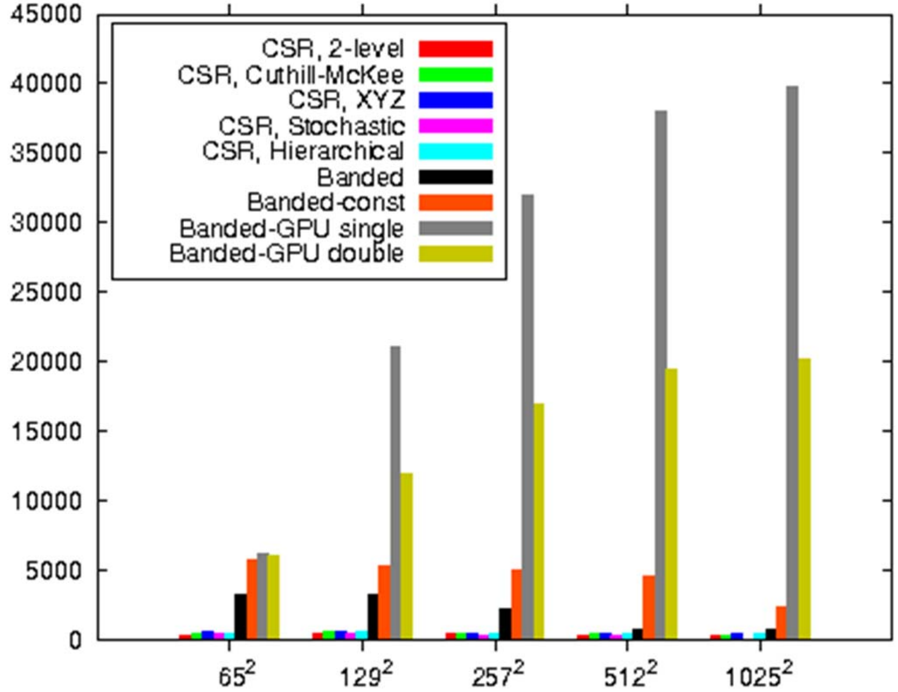


Sparse MV on TP Grids



0.1 – 0.7/2.4 GFLOP/s

Xeon E5450



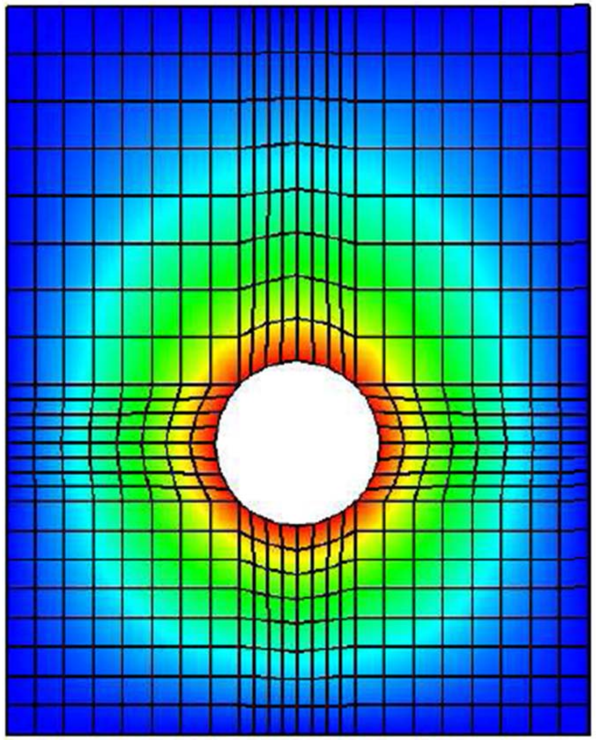
20 - 40 GFLOP/s

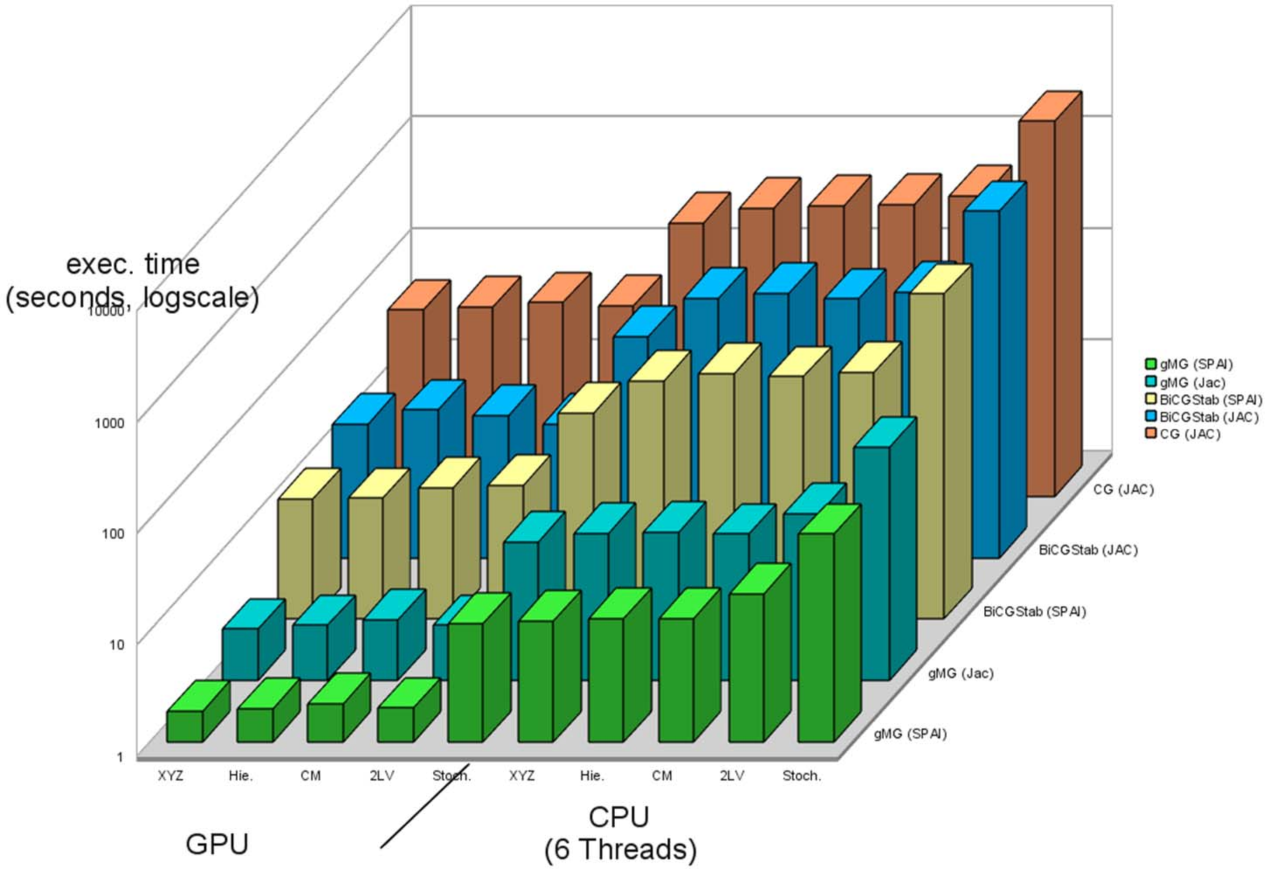
GeForce GTX 280

Poisson Solver Tests
(non TP grids)

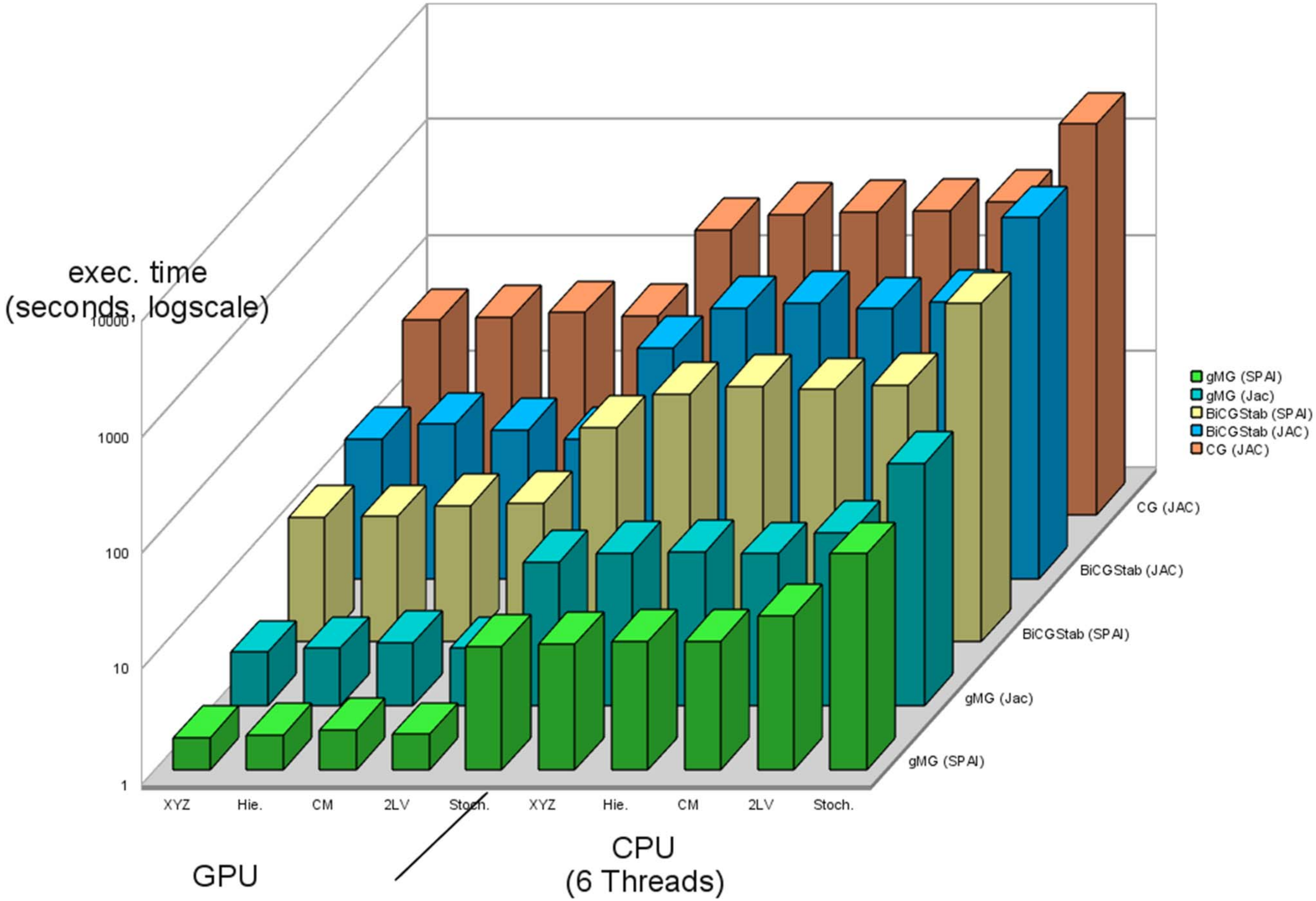
$$
\begin{aligned}
-\Delta u &= 0 \quad \text{in } \Omega, \\
u &= 0 \quad \text{on } \Gamma_1 \\
u &= 1 \quad \text{on } \Gamma_2
\end{aligned}
$$

	Q_1		Q_2	
L	N	non-zeros	N	non-zeros
4	576	4552	2176	32192
5	2176	18208	8448	128768
6	8448	72832	33280	515072
7	33280	291328	132096	2078720
8	132096	1172480	526336	8351744
9	526336	4704256	2101248	33480704
10	2101248	18845696	-	-





Poisson Solver Tests



Poisson Solver Tests

Identical solution, but differences of more than a

factor 1000x

regarding the CPU time for one „simple“ (small) subproblem

after „optimization“ on all levels!

HWON Challenges (I) – Basic Level

- **Strong ILU-like smoothers?**
  - ILU directly on GPUs?
  - SPAI – FSAI – AINV: Numerical properties?
  - Exploiting local structures: Linelet-GS, linewise GS-ADI?
  - 3D ???

- **Basic components for different FEM?**
  - Optimal numbering for nonconforming FEM?
  - FEM-adapted grid transfer via sparse MV?

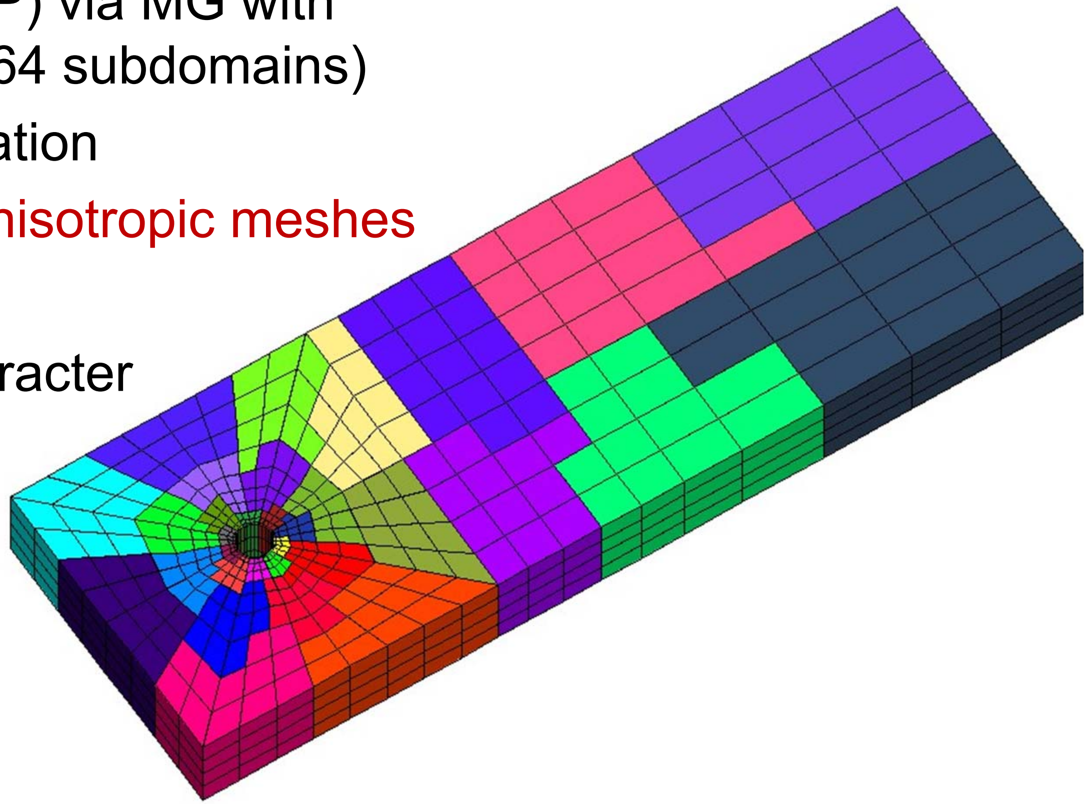
- **Realization of a FEM-gMG library**
  - BLAS-like: Generic vs. Hardware-optimized?

Parallel Performance

- Pressure Poisson Problem (PPP) via MG with **blockwise** ILU smoothing (1 – 64 subdomains)
  - Problems due to communication
  - Numerical problems w.r.t. anisotropic meshes

→ Increasing block-Jacobi character

→ ScaRC as hierarchically clustered recursive MG-DD solver



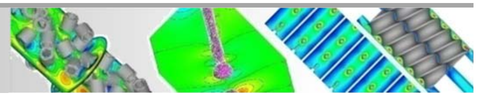
	1 P.	2 P.	4 P.	8 P.	16 P.	32 P.	64 P.
%Comm.	10%	24%	36%	45%	47%	55%	56%
# PPP-IT	2.2	3.0	3.9	4.9	5.2	5.7	6.2

HWON Challenges (II) – Advanced Level

- **Scalable (= robust & efficient) parallel solvers?**
  - Globally unstructured – locally structured
  - Exploit structured subdomains for scalable efficiency
  - Hide anisotropies locally to increase global robustness
  - Higher local arihtmetic costs, but less global communication

- **(Recursive) solver expert system?**
  – numerical + computational a priori knowledge!

- **Load balancing?**
  – due to ’total CPU time per accuracy per processor’?
  – dynamical a posteriori process?



HWON Challenges (III) – more Advanced Level

- **Adaptive meshing & complex (time dependent) geometries**
  - Grid Deformation: Flexible deformation & preserving logical structures
  - Fictitious Boundary Method as filter process for geometrical details

- **Coupling mechanisms**
  - Decoupled vs. Fully Coupled
  - Monolithic vs. Segregated

  → **Design new algorithms due to high arithmetic intensity**

CPU(Solver)	Method	#NT	Lift		Drag	
			mean	peak	mean	peak
14,358(81%)	Impl. MPSC	39	1%	1%	0%	2%
42,679(51%)	Semi-impl. DPM	165	0%	0%	0%	0%
64,485(54%)	Semi-expl. DPM	889	0%	8%	0%	0%

- **Higher order discretization in space and time**
  - Higher order time stepping schemes for increasing the solution part
  - Higher order FEM for more dense matrices

(→ talk by F. Schieweck & T.)

HWON Challenges (IV) – Benchmarking

- **How to define benchmarking scenarios which allow to measure the absolute performance???**

- **We have to consider absolute timings w.r.t. (virtually) optimal algorithms!**

HWON Summary: Extensive Tests show.....

- Even for `basic problems' (Poisson solver) the combination of numbering strategies + numerical components + hardware leads to differences in total efficiency of factor 1000x and more

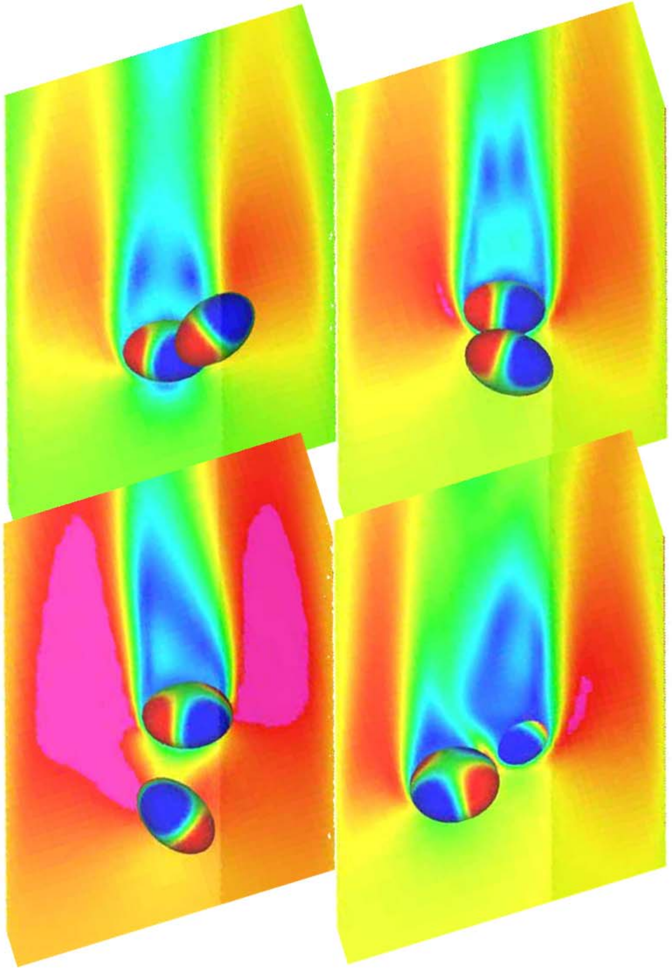
- `Parallel Peak Performance' with modern Numerics is even harder, already for moderate processor numbers

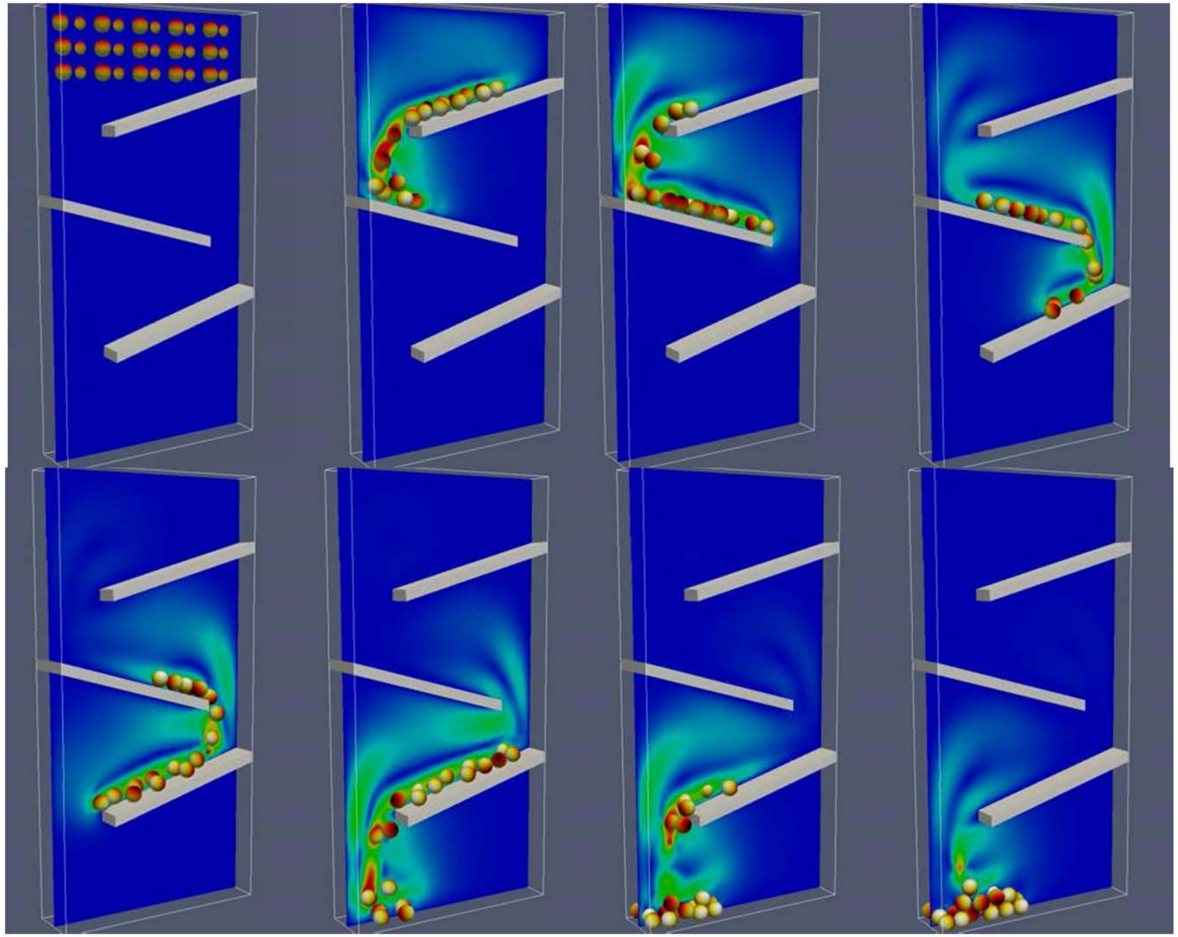
- Besides the mathematical part, the realization of flexible (and user-friendly?) mathematical software is very challenging

- **Absolute performance ratings are necessary!**

- Applying HWON to complex algorithms and applications is another story…

Application to Liquid-Solid Multiphase Flow

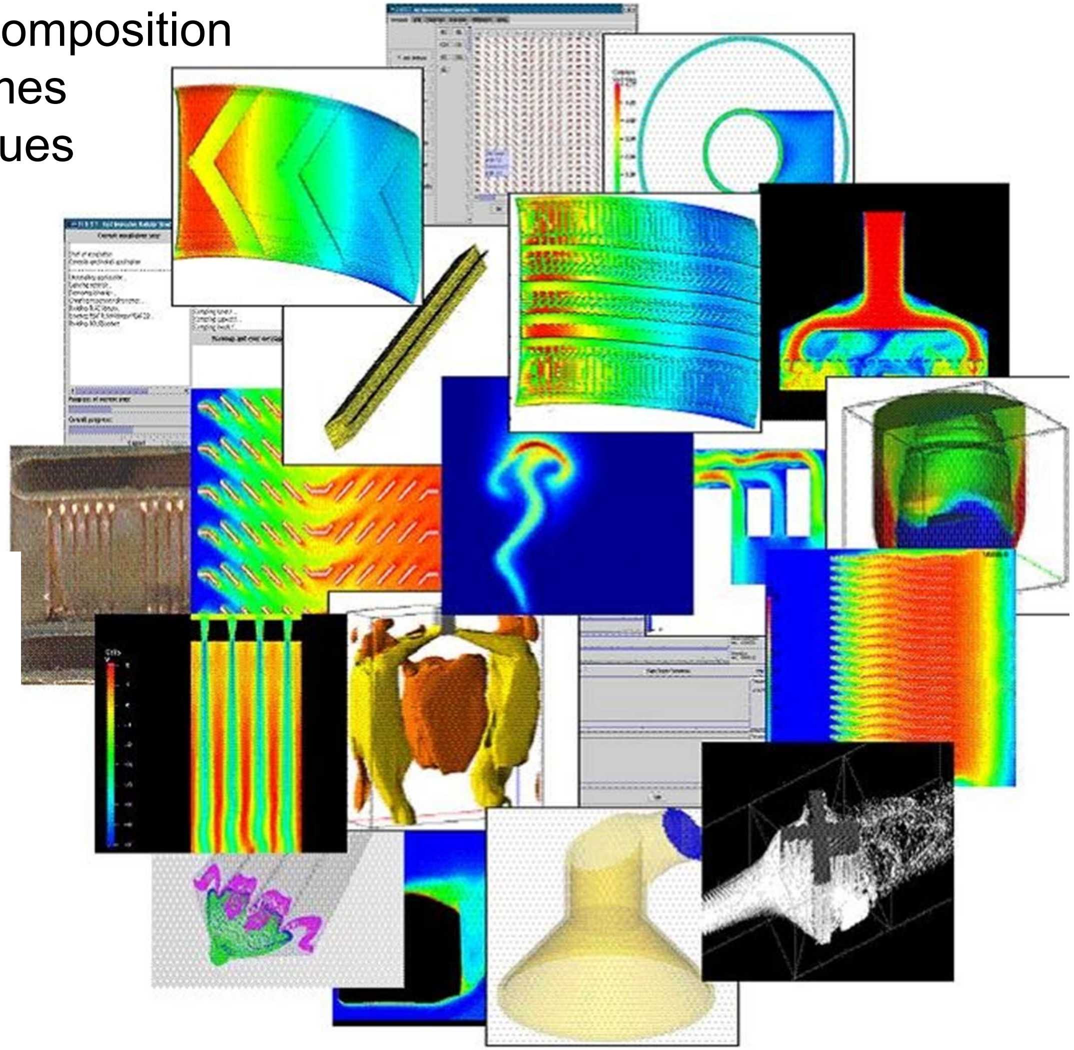




Basic Flow Solver: FeatFlow

<u>Numerical features:</u>

- Parallelization based on domain decomposition
- High order FEM discretization schemes
- FCT & EO FEM stabilization techniques
- Newton-Multigrid solvers
- Use of unstructured meshes
- Adaptive grid deformation



<u>HPC features</u>

- (Massively) parallel
- Soon: GPU computing
- Open source

Two phase flow (s-l) with resolved interphases

- **Fluid** motion is governed by the **Navier-Stokes equations**
- **Particle** motion is described by **Newton-Euler equations**

$$M_p \frac{dU_p}{dt} = F_p + F_{ex,col} + (\Delta M_p) g, \qquad I_p \frac{d\omega_p}{dt} = T_p - \omega_p \times (I_p \omega_p)$$

Hydrodynamic force — Torque

$$F_p = -\int_{\Gamma_p} \sigma \cdot n_p \, d\Gamma_p \qquad T_p = -\int_{\Gamma_p} (X - X_p) \times (\sigma \cdot n_p) \, d\Gamma_p$$

Postprocessing the actual flow field

Fictitious Boundary Method

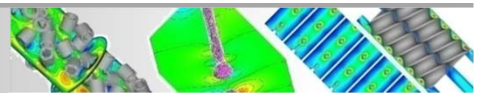
- Surface integral is replaced by volume integral
- Use of monitor function (liquid/solid)

$$\alpha_p(X) = \begin{cases} 1 & \text{for} \quad X \in \Omega_p \\ 0 & \text{for} \quad X \in \Omega_f \end{cases}$$

- Normal to particle surface vector is non-zero only at the surface of particles $n_p = \nabla \alpha_p$

$$F_p = -\int_{\Gamma_p} \sigma \cdot n_p \, d\Gamma_p = -\int_{\Omega_T} \sigma \cdot \nabla \alpha_p \, d\Omega_T$$

$$T_p = -\int_{\Gamma_p} (X - X_p) \times (\sigma \cdot n_p) \, d\Gamma_p = -\int_{\Omega_T} (X - X_p) \times (\sigma \cdot \nabla \alpha_P) \, d\Omega_T$$



Two phase flow (s-l) with resolved interphases

Fictitious Boundary Method

For computed
$U_p^{n+1}, \omega_p^{n+1}$

→

Position update:
$$\frac{dX_p}{dt} = U_p,$$

Angle update:
$$\frac{d\theta_p}{dt} = \omega_p$$

→

$X_p^{n+1}, \theta_p^{n+1}$

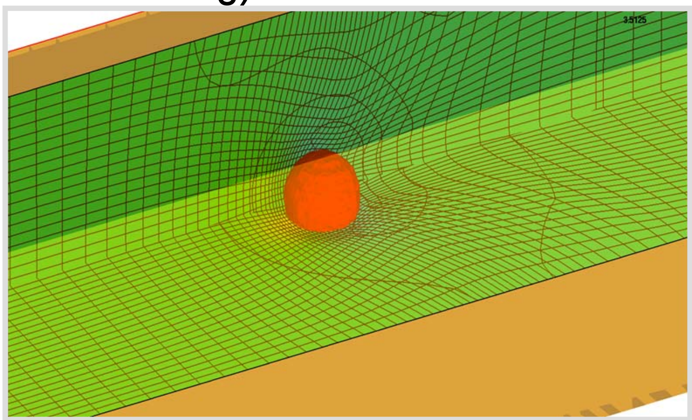
Velocity “boundary condition” imposed for particles:

$$u(X) = U_p + \omega_p \times (X - X_p)$$

- supports HPC concepts (constant data structures, optimal load balancing)
- reduces requirements put on the computational mesh
- relatively low resolution

→

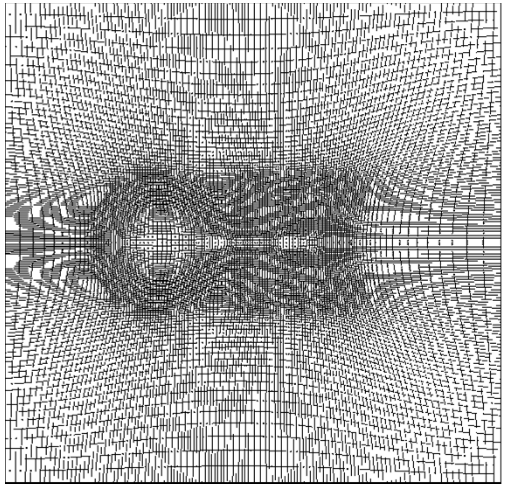
- Brute force → Finer mesh resolution
- High resolution interpolation functions
- **Grid deformation** (+ monitor function)

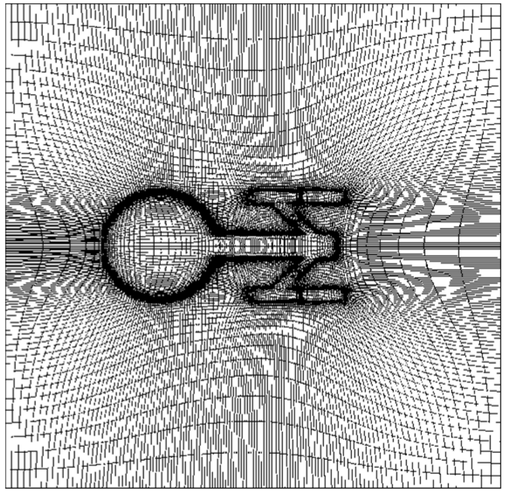


Grid Deformation Method

Idea : construct transformation ϕ, $x = \phi(\xi, t)$ with $\det \nabla \phi = f$

⟹ local mesh area $\approx f$





1. Compute monitor function $f(x,t) > 0,\ f \in C^1$
and

$$\int_\Omega f^{-1}(x,t)dx = |\Omega|, \quad \forall t \in [0,1]$$

2. Solve$(t \in [0,1])$

$$\Delta v(\xi, t) = -\frac{\partial}{\partial t}\left(\frac{1}{f(\xi, t)}\right), \quad \left.\frac{\partial v}{\partial n}\right|_{\partial\Omega} = 0$$

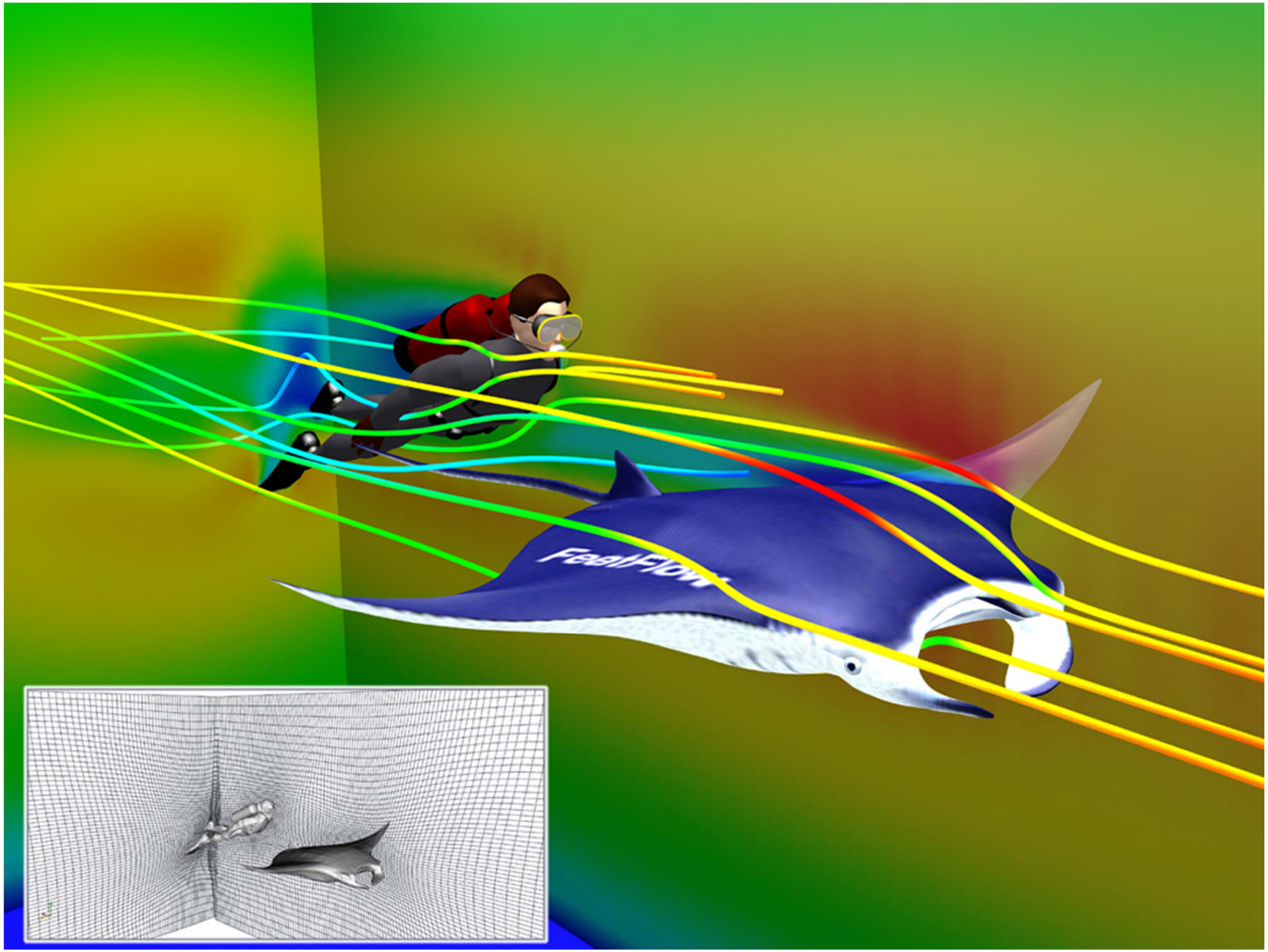
3. Solve the ODE system

$$\frac{\partial}{\partial t}\phi(\xi, t) = f(\phi(\xi, t), t)\nabla v(\phi(\xi, t), t)$$

new grid points: $x_i = \phi(\xi_i, 1)$

Grid deformation preserves the (local) logical structure of the grid

Generalized Tensorproduct Meshes



Operator-Splitting Approach

The algorithm for $t^n \rightarrow t^{n+1}$ consists of the following 4 substeps

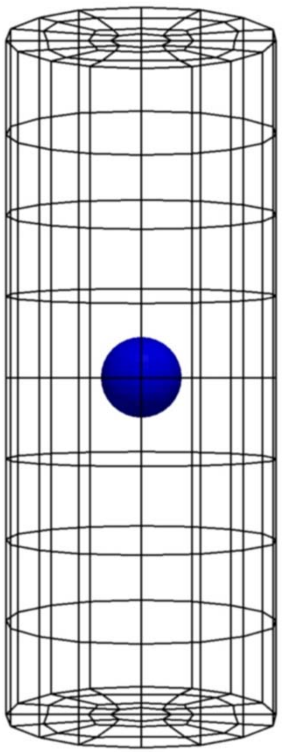
1. Fluid velocity and pressure: $NSE\left(u_f^{n+1}, p^{n+1}\right) = BC\left(\Omega_p^n, u_p^n\right)$
2. Calculate hydrodynamic forces: F_p^{n+1}
3. Calculate velocity of particles: $u_p^{n+1} = g\left(F_p^{n+1}\right)$ (collision model)
4. Update position of particles: $\Omega_p^{n+1} = f\left(u_p^{n+1}\right)$
5. Align new mesh

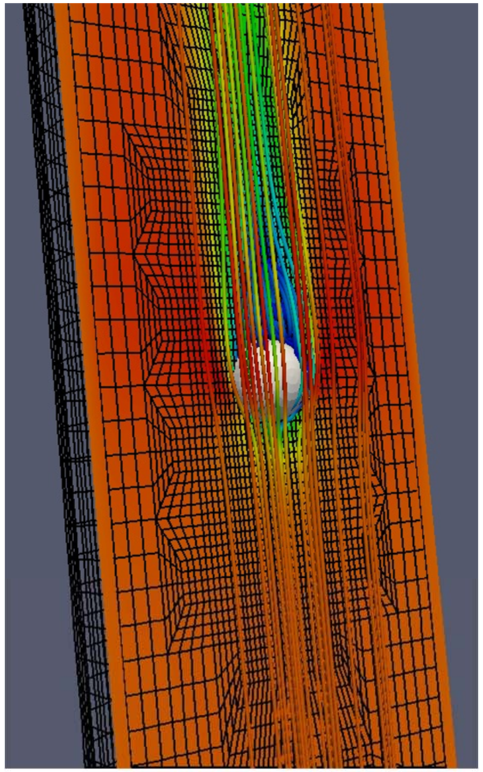
→ Required: efficient calculation of hydrodynamic forces
→ Required: efficient treatment of particle interaction (?)
→ Required: fast (nonstationary) Navier-Stokes solvers

Benchmarking and Validation

Free fall of particles:

- Terminal velocity
- Different physical parameters
- Different geometrical parameters





Münster, R.; Mierka, O.; Turek, S.: Finite Element fictitious boundary methods (FEM-FBM) for 3D particulate flow, IJNMF, 2010, accepted

$d_s = 0.3, \quad \rho_s = 1.14$

ν	$U_{featflow}$	U_{exp}	Relative error (%)
0.02	5.885	6.283	6.33
0.05	4.133	3.972	4.05
0.1	2.588	2.426	6.66
0.2	1.492	1.401	6.50

$d_s = 0.2, \quad \rho_s = 1.14$

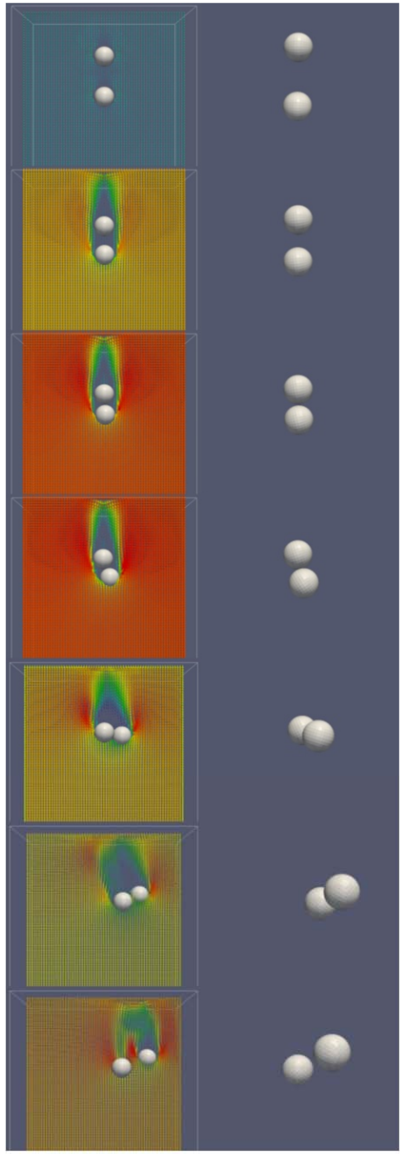
ν	$U_{featflow}$	U_{exp}	Relative error (%)
0.02	4.370	4.334	0.83
0.05	2.699	2.489	8.44
0.1	1.649	1.552	6.25
0.2	0.946	0.870	8.74

$d_s = 0.3, \quad \rho_s = 1.02$

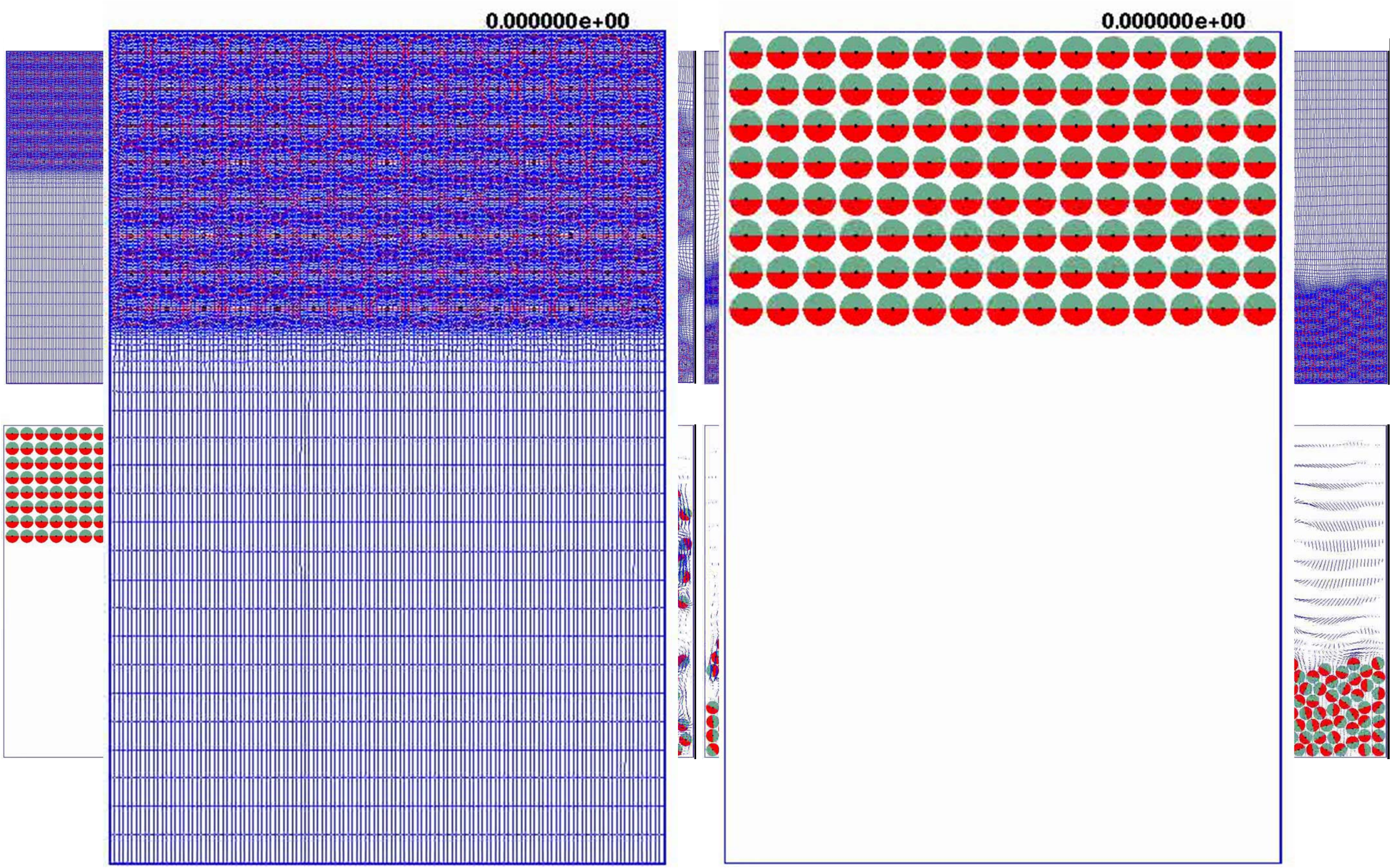
ν	$U_{featflow}$	U_{exp}	Relative error (%)
0.01	2.167	2.107	2.84
0.02	1.495	1.436	4.11
0.05	0.809	0.749	8.01
0.1	0.402	0.404	0.44
0.2	0.218	0.216	1.02

$d_s = 0.2, \quad \rho_s = 1.02$

ν	$U_{featflow}$	U_{exp}	Relative error (%)
0.01	1.4660	1.4110	3.90
0.02	0.9998	0.9129	9.52
0.05	0.4917	0.4603	6.82
0.1	0.2637	0.2571	2.57
0.2	0.1335	0.1317	1.37



Sedimentation of Many Particles



Repulsive Force Collision Model

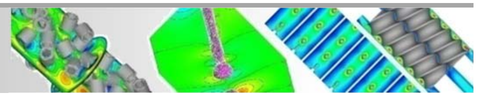
- Handling of small gaps and contact between particles
- Dealing with overlapping in numerical simulations

For the particle-particle collisions (analogous for the particle-wall collisions), the repulsive forces between particles read:

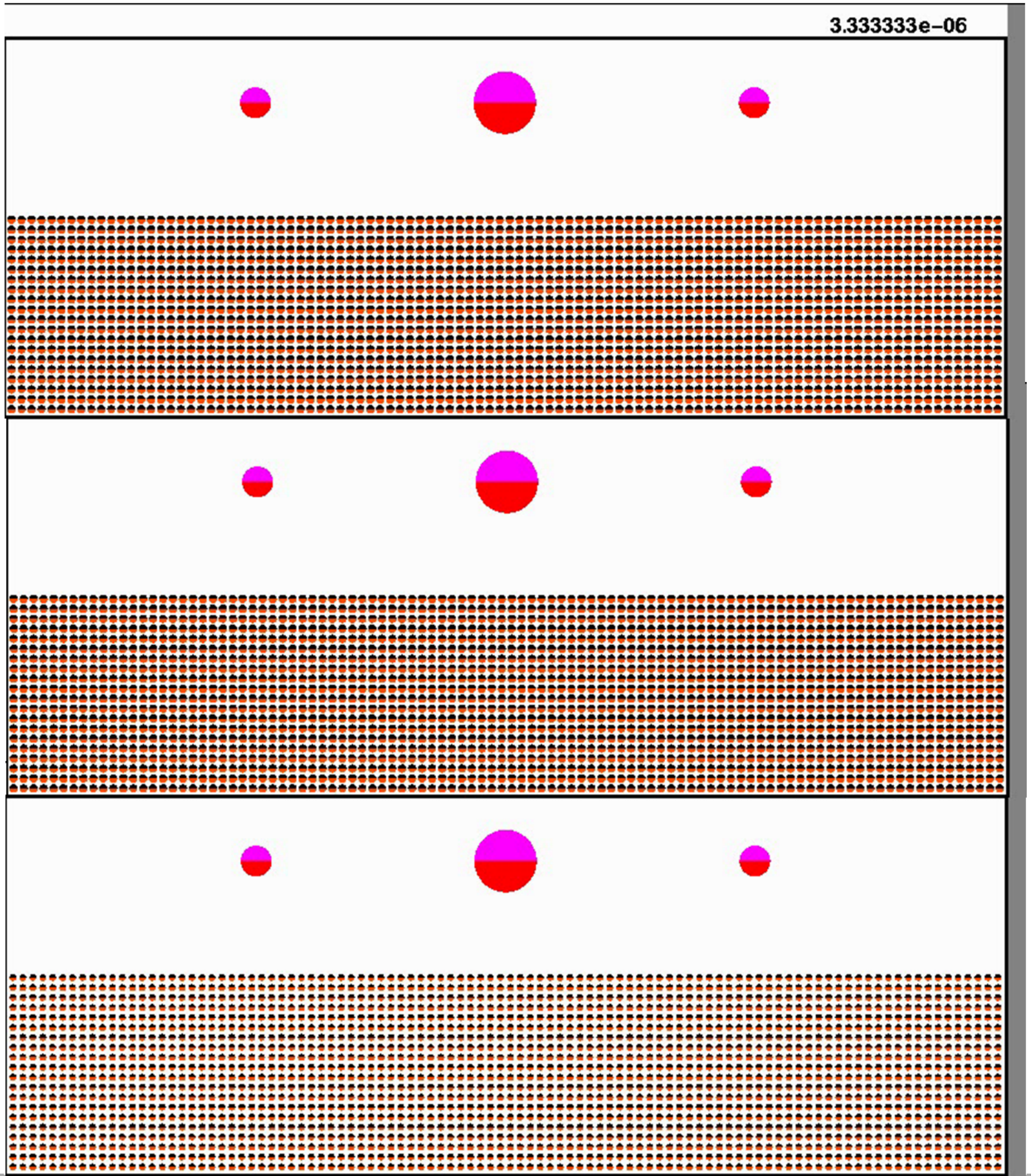
$$F_{ij}^{P} = \begin{cases} 0 & \text{for} \quad d_{i,j} > R_i + R_j + \rho \\ \dfrac{1}{\varepsilon_P}\left(X_i - X_j\right)\left(R_i + R_j + \rho - d_{i,j}\right)^2 & \text{for} \quad R_i + R_j \le d_{i,j} \le R_i + R_j + \rho \\ \dfrac{1}{\varepsilon_P^{'}}\left(X_i - X_j\right)\left(R_i + R_j - d_{i,j}\right) & \text{for} \quad d_{i,j} < R_i + R_j \end{cases}$$

The total repulsive forces exerted on the i-th particle by the other particles and the walls can be expressed as follows:

$$F_i^{'} = \sum_{j=1,\, j \neq i}^{N} F_{i,j}^{P} + F_i^{W}$$



Impact of heavy balls on small particles



$\rho_f = 1$

$\rho_{bd} = 2$

$\rho_{sp} = 20$

$\rho_f = 1$

$\rho_{bd} = 2$

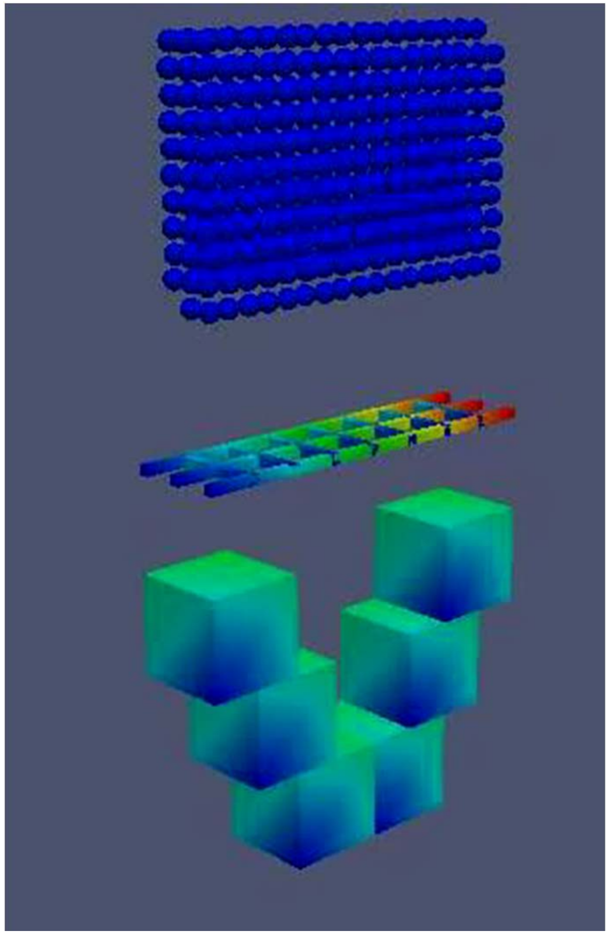
$\rho_{sp} = 2$

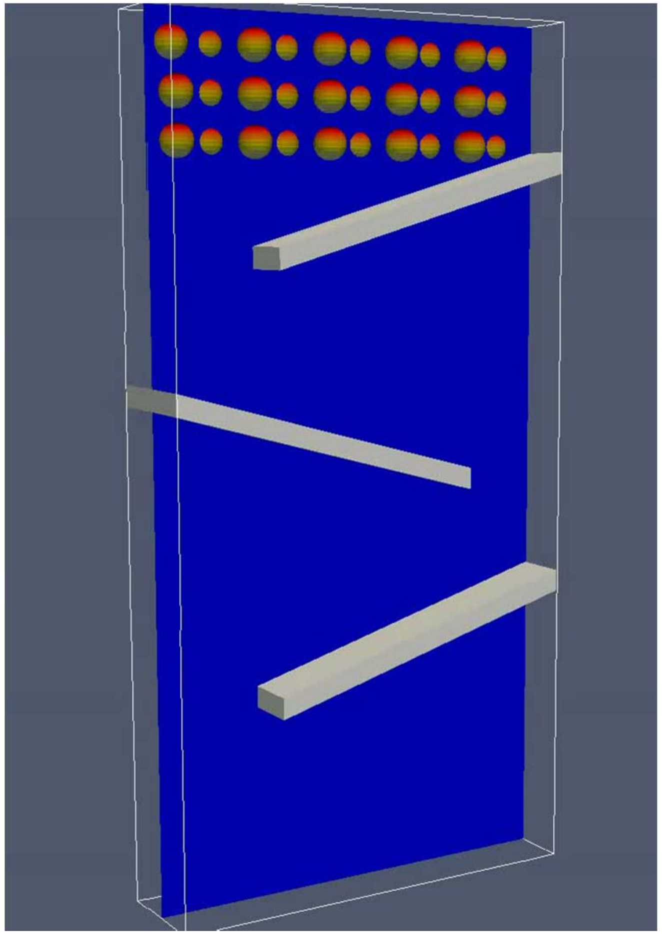
$\rho_f = 1$

$\rho_{bd} = 2$

$\rho_{sp} = 1.1$

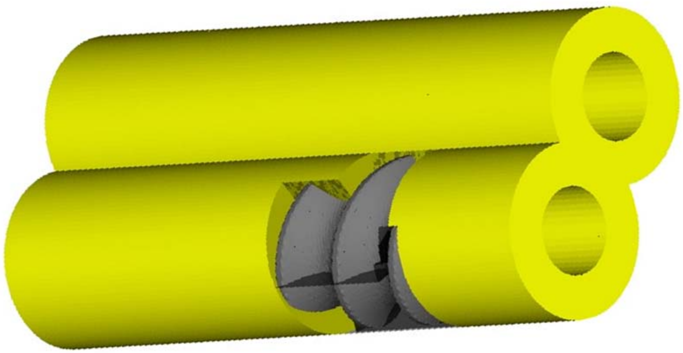
Sedimentation of particles in a complex 3D domain

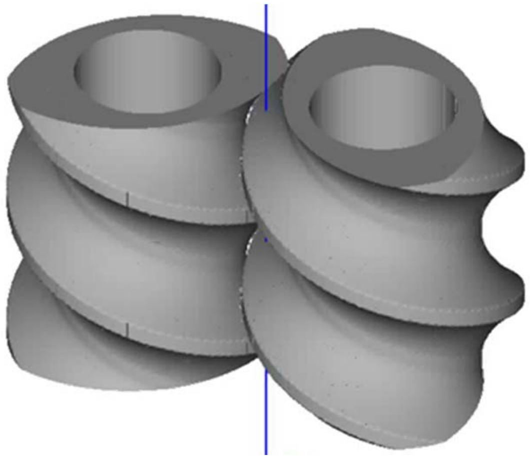




Twinscrew Flow Simulations

Geometrical representation of the twinscrews → Fictitious Boundary Method



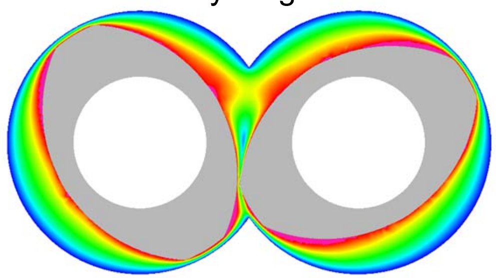


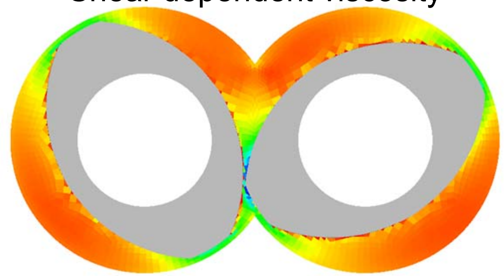
- **Fast and accurate description of rotating geometry**
- Applicable for conveying and kneading elements
- Mathematical description available for single, double- or triplet-flighted screws
- Surface and body of the screws are known at any time
- Mathematical formulation replaces external CAD-description
- **Non-Newtonian and temperature dependent** physical properties including **rigid particles**
- Heat dissipation due to high shear rates

In cooperation with:

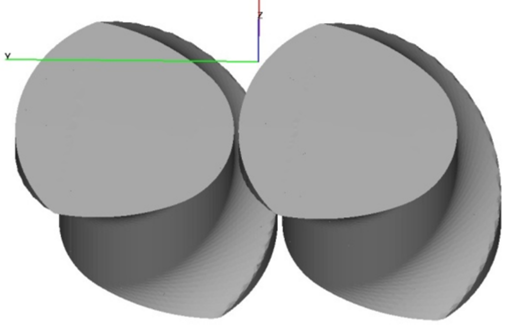
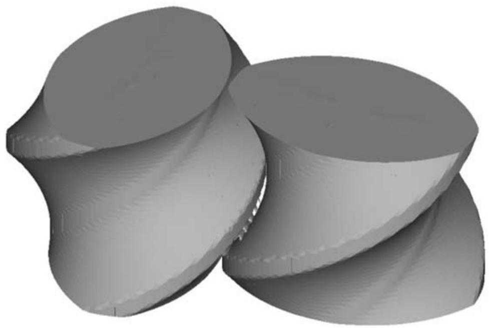
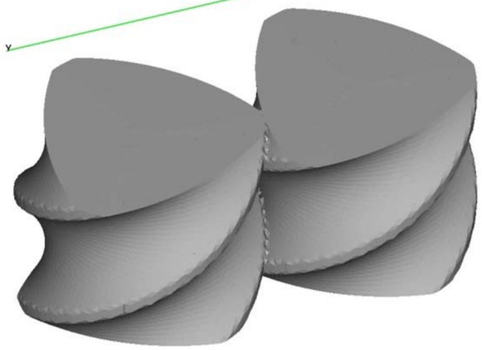
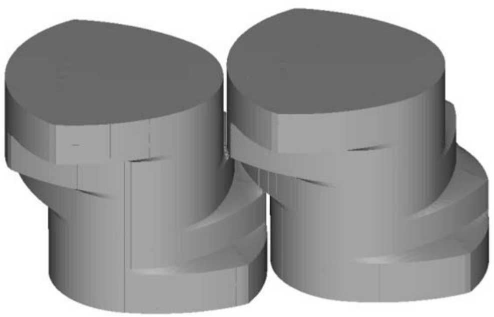
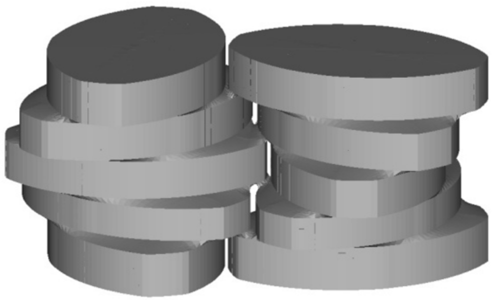
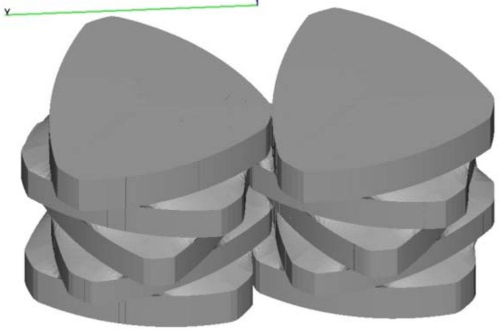
UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

IANUS Simulation



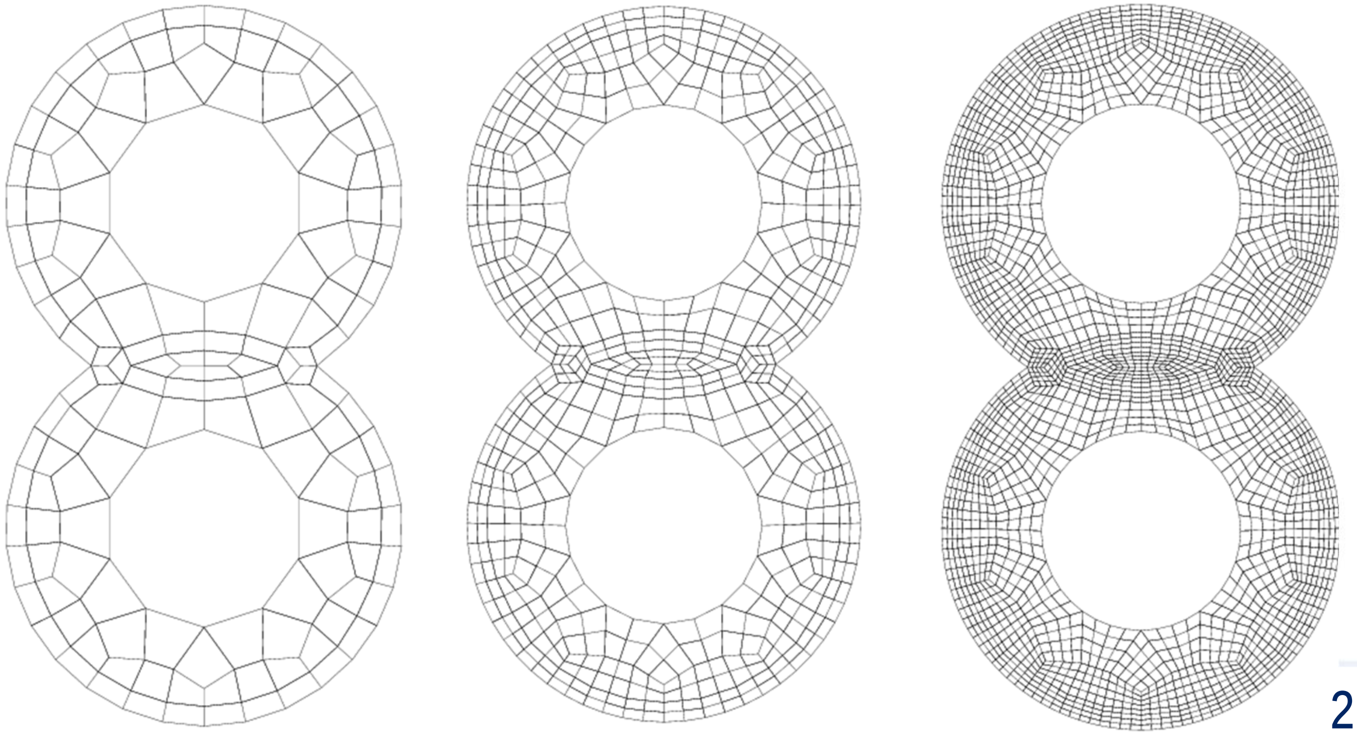


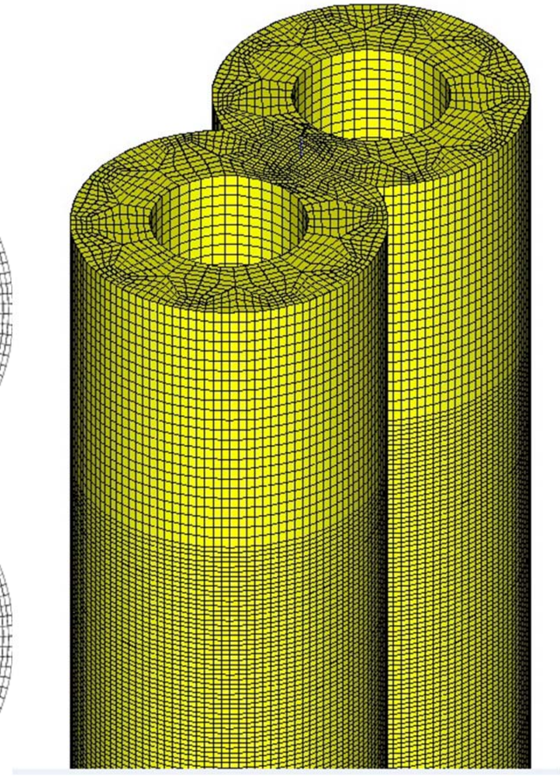
Library of Conveying and Mixing Elements

	1 flighted	2 flighted	3 flighted
conveying			
mixing			

Static Mesh Refinement & Dynamic FBM

level 1 level 2 level 3

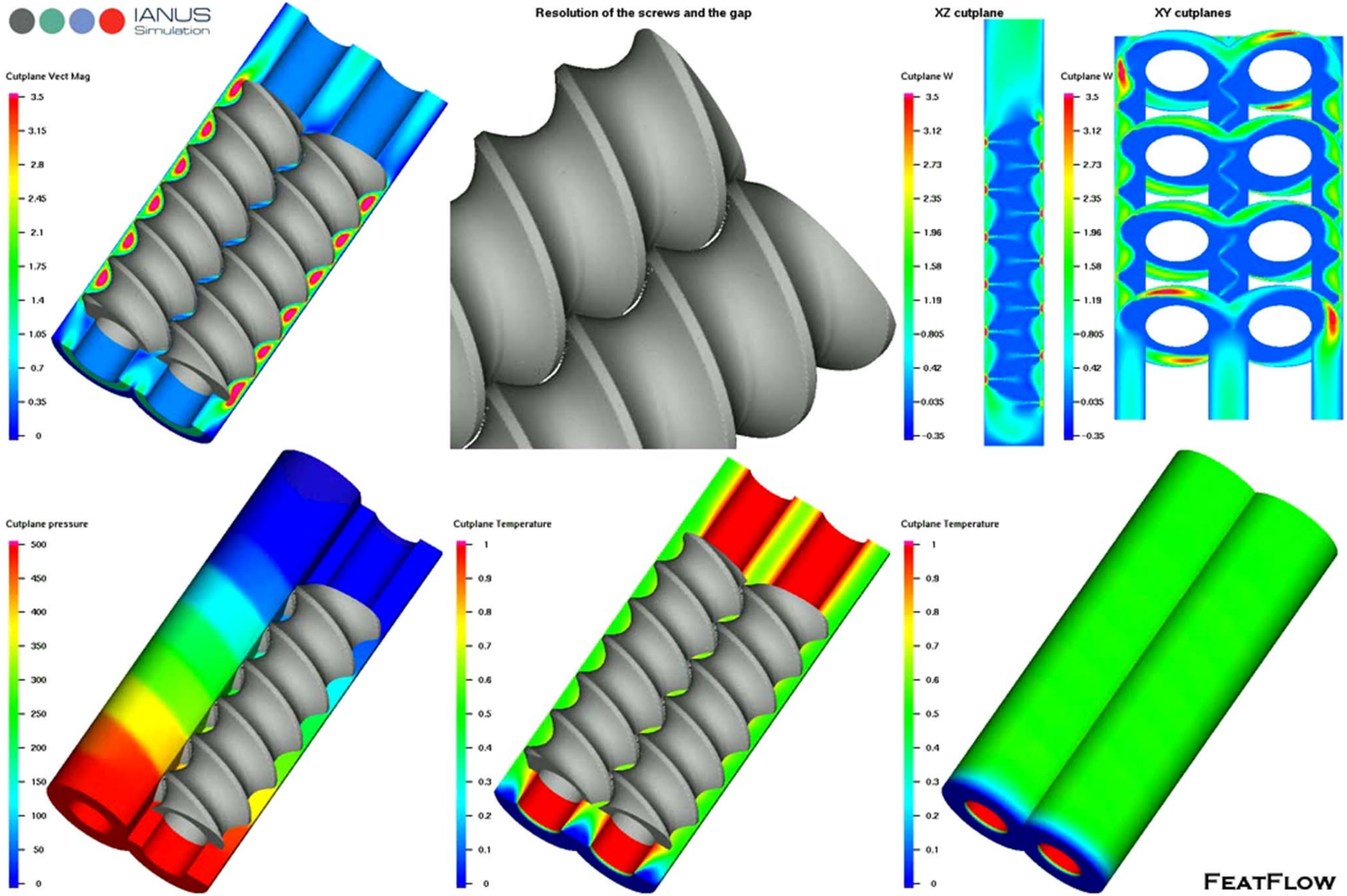




2D mesh extrusion into 3D

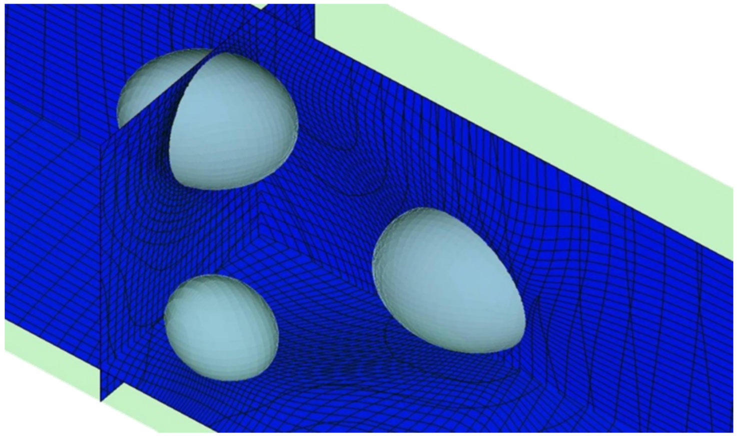
Pre-refined regions in the vicinity of gaps

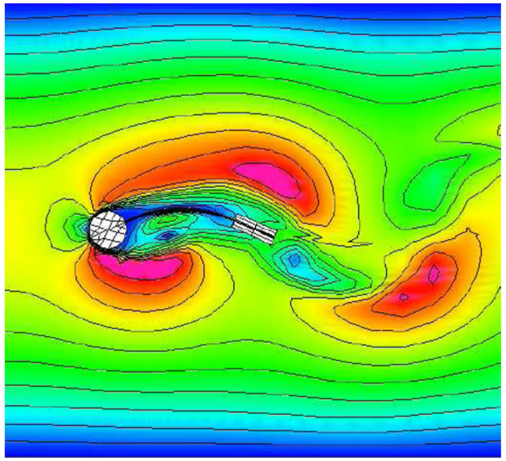
Twinscrew Flow Simulations

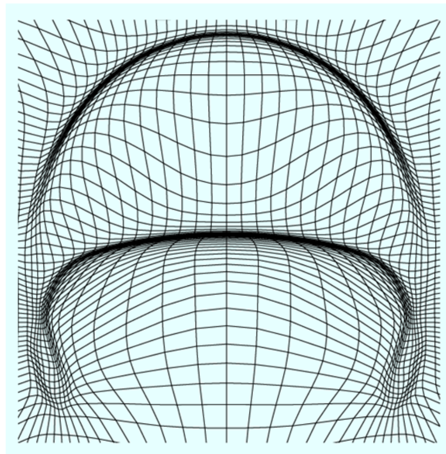


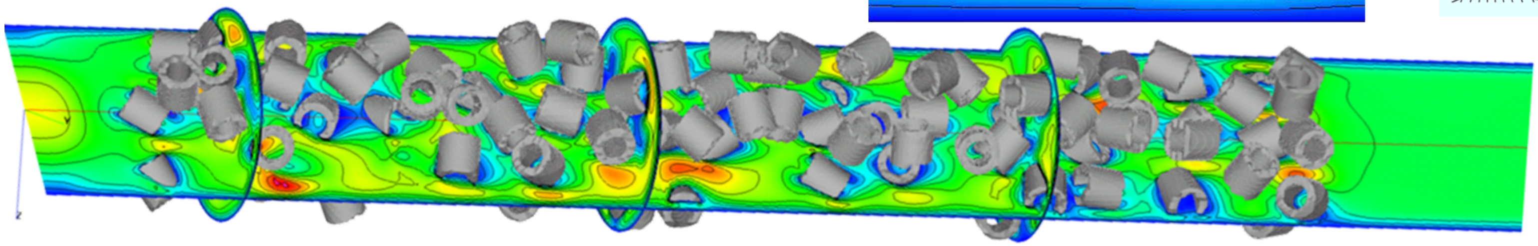
Next Steps for Liquid-Solid Multiphase Flow

- Adaptive time stepping + adaptive grid alignment/ALE.
- Coupling with turbulence models.
- Deformable particles/fluid-structure interaction.
- Analysis of viscoelastic effects.
- Benchmarking and experimental validation for many particles.









Some HWON Rules of Thumb

- Realize all MG components via sparse MV (preconditioners, grid transfer) & Optimize sparse MV w.r.t. FEM space, numbering and hardware

→ **Generic and hardware-optimized `gMG-FEM-BLAS' Toolbox**

- Use higher order in time (large time steps) + space (large FEM stencils)

→ **High arithmetic intensity via dominant `solution part' (→ gMG)**

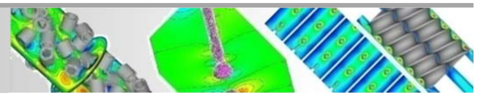
- Design strongly coupled schemes (globally) with Operator-Splitting components (locally)

→ **Combine (outer) high robustness & (inner) high efficiency**

- Exploit locally regular structures to improve global convergence

→ **Strong local solvers cost nothing & Hide irregularities locally**

→ **Patchwise adaptivity, generalized TP meshes, Grid Deformation, FBM,…**



Conclusion: Huge Potential for the Future …

However:

- **Numerical Simulation & High Performance Computing** have to consider recent and future hardware trends, particularly for heterogeneous multicore architectures and massively parallel systems!

- The combination of **'Hardware-oriented Numerics'** and special **'Data Structures/Algorithms'** and **'Unconventional Hardware'** has to be used!

…or many of the existing (academic/commercial) PDE software packages will be 'worthless' in a few years!